Vorrichtung zur photorealistischen Darstellung von dynamischen komplexen dreidimensionalen Szenen mittels des Ray-Tracing Verfahrens

Die Erfindung betrifft eine Vorrichtung, mit der dynamische, komplexe dreidimensionale Szenen mit hohen Bildwiederholraten unter Verwendung einer Echtzeit Ray-Tracing Hardwarearchitektur auf einem zweidimensionalen Display dargestellt werden können. Dynamische Szenen sind Szenen, in denen sich neben der Kameraposition auch die Geometrie der darzustellenden Objekte von Frame zu Frame ändern kann. Die Erfindung zeichnet sich vor allem dadurch aus, dass sie eine Hierarchiestruktur von Objekten unterstützt, das heißt die Hauptszene kann aus mehreren Objekten bestehen, die jeweils aus weiteren Objekten aufgebaut sind, wobei diese Schachtelung beliebig fortgeführt werden kann. Die auf den einzelnen Hierarchieebenen befindlichen Objekte können sowohl einzeln, als auch im Verbund bewegt werden. Hierdurch ist es möglich, komplexe Szenen mit hoher Dynamik zu erstellen und durch Verwendung des gleichen Objektes an mehreren Stellen der Szene die Repräsentation der Szene im Speicher klein zu halten.

Zur erfindungsgemäßen Realisierung dieser verschachtelten Objektebenen, wird die hardwaremäßige Umsetzung der bekannten Ray-Tracing-Pipeline um eine hardwaremäßig realisierte Transformationseinheit erweitert, welche die Strahlen in die Objekte hinein transformiert. Diese Einheit ist zur optimalen Ausnutzung der Hardwareressourcen nur einmal vorhanden und wird neben der Objektraumtransformation auch zur Berechnung des Schnittpunktes des Strahles mit einem Dreieck, der Erstellung von Primärstrahlen und zur Erstellung von Sekundärstrahlen verwendet.

Durch die Verwendung von speziellen auf Ray-Tracing ausgelegten Prozessoren gestattet die Erfindung dem Anwender die volle Programmierbarkeit des Systems, indem erfindungsgemäß eine neuartige Prozessorarchitektur, bestehend aus der Kombination eines Standard Prozessorkerns mit einem, oder mehreren speziellen Ray-Tracing Befehlen verwendet wird. Die Verwendung dieser Ray-Tracing Prozessoren erlaubt die Programmierung verschiedenster Ray-Tracing-Verfahren. Primitive Objekte der Szene können programmierbar gestaltet werden, so dass im Gegensatz zu heutigen Grafikkarten auch die Verwendung von Spline Flächen möglich ist, indem ein spezieller

Schnittpunktberechnungsalgorithmus eines Strahles mit der Splinefläche programmiert wird. Wie in heutiger Rasterisierungshardware üblich können verschiedenste Shadingmodelle für Oberflächen programmiert werden.

Zur Ausgabe der Bilddaten auf einem Display kann die Erfindung durch die Verwendung gemeinsamer Frame-Buffer und Z-Buffer mit einer dem Stand der Technik entsprechenden Rasterisierungshardware kombiniert werden.

Stand der Technik

Der Stand der Technik bezüglich der Darstellung von dreidimensionalen Szenen ist derzeit in zwei Haupt-Sektoren einzuteilen, das Rasterisierungsverfahren und das Ray-Tracing Verfahren (siehe Computer Graphics / Addison-Wesley ISBN 0201848406).

Das bekannte, vor allem in Computer-Graphikkarten zum Einsatz kommende Rasterisierungsverfahren beruht auf dem Prinzip, jede Geometrie der Szene auf einen Frame-Buffer und Z-Buffer zu projizieren. Hierzu werden die Farb- und Helligkeitswerte der Pixel im Frame-Buffer und die geometrischen Tiefenwerte im Z-Buffer gespeichert, jedoch nur dann, wenn der vorherige geometrische Wert im Z-Buffer größer (weiter vom Betrachter entfernt liegend) als der neue ist. Hierdurch wird sichergestellt, dass nähere Objekte fernere überschreiben und nach Ablauf des Verfahrens nur noch die wirklich sichtbaren Objekte im Frame-Buffer abgebildet sind.

Dieses Verfahren hat jedoch den entscheidenden Nachteil, dass aufwändige Szenen mit Millionen von Objekten mit der bisher bekannten Hardware nicht in Echtzeit dargestellt werden können, da es in der Regel erforderlich ist, alle Dreiecke (Objekte) der Szene zu projizieren. Des weiteren wird ein Frame-Buffer und Z-Buffer benötigt, auf dem viele Milliarden Schreiboperationen in der Sekunde durchgeführt werden müssen, wobei zum Bildaufbau die meisten Pixel mehrfach pro Frame überschrieben werden. Das Überschreiben des zum Betrachter weiter entfernten Pixels durch Pixel näherer Objekte hat zur Folge, dass bereits berechnete Daten verworfen werden, wodurch eine optimale Systemleistung nicht realisiert werden kann.

Schatten können mit aufwändigen Techniken auf heutiger Rasterisierungshardware berechnet werden, jedoch ergeben sich bei komplexen Szenen Probleme hinsichtlich der

Genauigkeit. Spiegelungen an gekrümmten Flächen, sowie die Berechnung von Lichtbrechungen sind mit dieser Technik nicht physikalisch korrekt zu realisieren.

Leistungssteigernde Verbesserung schafft ein zweites Verfahren, das Ray-Tracing Verfahren, das durch seine photorealistischen Bilder, aber auch durch die Rechenkomplexität bekannt ist. Die Grundidee von Ray-Tracing steht in nahem Bezug zu physikalischen Lichtverteilungsmodellen (siehe Computer Graphics / Addison-Wesley ISBN 0201848406).

In einer realen Umgebung wird Licht von Lichtquellen emittiert und nach physikalischen Gesetzen in der Szene verteilt. Mit einer Kamera kann das Bild der Umgebung eingefangen werden. Ray-Tracing geht den umgekehrten Weg und verfolgt das Licht von der Kamera, welche die Betrachterposition darstellt, zurück zu ihrer Quelle. Hierzu wird für jedes Pixel des Bildes ein virtueller Strahl in die, das Pixel beleuchtende, Richtung geschossen. Dieses Schießen des Strahles nennt man Ray-Casting. Trifft der Strahl ein Objekt so berechnet sich die Farbe des Pixels unter anderem aus der Farbe des getroffenen Objektes, der Oberflächennormalen und den vom Auftreffpunkt sichtbaren Lichtquellen. Die sichtbaren Lichtquellen sind durch die Verfolgung der Sekundärstrahlen, die von jeder Lichtquelle zu dem Auftreffpunkt geschossen werden, zu ermitteln. Treffen diese Schattenstrahlen ein Objekt zwischen Lichtquelle und Auftreffpunkt, so liegt der Punkt im Schatten bezüglich der Lichtquelle.

Dieses Verfahren ermöglicht neben der beschriebenen Schattenberechnung auch noch die Berechnung von Spiegelungen und der Brechungen des Lichtes, indem Reflexionsstrahlen bzw. gebrochene Sekundärstrahlen berechnet werden. Des weiteren können Szenen nahezu beliebiger Größe gehandhabt und dargestellt werden. Der Grund hierfür liegt in der Anwendung einer Beschleunigungsstruktur. Dies ist ein spezielles Verfahren mit entsprechender Datenstruktur, die es ermöglicht, den virtuellen Strahl schnell durch die Szene zu „schießen" bzw. zu traversieren. Auf dem Weg werden einige Objekte selektiert, die mögliche Trefferkandidaten sind, wodurch der Auftreffpunkt schnell gefunden wird. Theoretische Untersuchungen haben ergeben, dass die Komplexität des Ray-Tracing Verfahrens im Mittel logarithmisch mit der Szenengröße wächst. Das heißt, eine Quadrierung der Zahl der Objekte der Szene bedeutet lediglich den doppelten Rechenaufwand.

Typische Beschleunigungsstrukturen sind zum Beispiel das regelmäßige Grid, der k-D Baum, der Octree und die Bounding-Volume-Hierarchy (siehe Computer Graphics / Addison-Wesley ISBN 0201848406). Allen diesen Verfahren liegt die Idee zu Grunde, den

Raum in viele Teilräume aufzuteilen und für jeden solchen Teilraum die dort vorhandene Geometrie zu speichern. Das Traversierungsverfahren verfolgt den Strahl dann von Teilraum zu Teilraum und schneidet diesen immer mit genau den Objekten, die sich in dem Teilraum befinden. Die vier Verfahren unterscheiden sich nur in der Anordnung der Teilräume. Bei dem regelmäßigen Grid ist der Raum in würfelförmige Teilräume gleicher Größe unterteilt. Zur Veranschaulichung wird in diesem Zusammenhang auf die Darstellung der Figur 7 verwiesen. Die drei anderen Verfahren beruhen auf einer rekursiven Unterteilung des Raumes. Beim k-D Baum Verfahren wird der Startraum rekursiv an einer beliebigen Stelle achsenparallel geteilt. Hierzu wird zur Veranschaulichung auf Figur 8 verwiesen. Diese Aufteilung des Raumes wird in einer rekursiven Datenstruktur (einem binären Baum) gespeichert. Das dritte Verfahren namens Octree, ist ebenfalls rekursiv, nur werden die betrachteten Teilräume immer in 8 gleichgroße rechteckige Teilräume unterteilt. Zur Veranschaulichung wird hierzu auf Figur 9 verwiesen. Die Bounding-Volume-Hierarchie unterteilt den Raum in n beliebige Volumina, die sogar überlappen dürfen, was bei den anderen Verfahren nicht erlaubt ist.

Im Gegensatz zum Rasterisierungsverfahren existiert momentan keine reine Hardwarelösung, die das Ray-Tracing-Verfahren umsetzt, sondern nur softwarebasierende Systeme, die vergleichsweise viel Rechenleistung und Rechenzeit erfordern. Zur Veranschaulichung des zeitlichen Umfanges der Berechnungen sei bemerkt, dass abhängig von der Komplexität des Bildes und der verwendeten Software mit der momentan dem Stand der Technik entsprechenden PC Hardware eine Rechenzeit von einigen Sekunden bis zu mehreren Stunden benötigt wird, um ein einzelnes Standbild nach diesem Verfahren zu erstellen. Die Berechnungen von Bewegtbildern erfordert entsprechend viel Rechenzeit und/oder die Verfügbarkeit von speziellen Großrechnern.

Der Lehrstuhl Computergraphik an der Universität des Saarlandes hat ein softwarebasiertes Echtzeit Ray-Tracing System entwickelt, das auf einem Cluster von über 20 Rechnern zum Einsatz kommt.

Im US Patent 6,597,359 B1 ist eine Hardwarelösung für das Ray-Tracing Verfahren beschrieben, der sich jedoch auf statische Szenen beschränkt.

Das US Patent 5,933,146 beschreibt ebenfalls eine Hardwarelösung für das Ray-Tracing-Verfahren, auch eingeschränkt auf statische Szenen.

Das Paper „SaarCOR - A Hardware Architecture for Ray-Tracing" vom Lehrstuhl Computergraphik der Universität des Saarlandes beschreibt eine Hardwarearchitektur für Ray-Tracing jedoch wiederum limitiert auf statische Szenen.

Das Paper „A Simple and Practical Method for Interactive Ray-Tracing of Dynamic Scenes" vom Lehrstuhl Computergraphik der Universität des Saarlandes beschreibt einen Softwareansatz zur Unterstützung von dynamischen Szenen in einem Ray-Tracer. Das beschriebene Software Verfahren verwendet jedoch nur eine Stufe von Objekten, kann also keine Schachtelung in mehreren Stufen durchführen.

Der beschriebene Stand der Technik bietet momentan weder Software- noch Hardwarelösungen, mit denen komplexe dynamische Szenen in Echtzeit dargestellt werden können. Bei den bekannten Rasterisierungsverfahren liegt die Leistungsbegrenzung in der Zahl der darzustellenden Objekte.

Ray-Tracing Systeme können zwar viele Dreiecke darstellen, sind jedoch wegen der benötigten Vorberechnungen darin beschränkt, dass die Position nur eingeschränkt geändert werden kann. Szenen aus einigen Milliarden Dreiecken erfordern sehr viel Rechenleistung und Speicher und sind nur auf schnellen und komplexen Großrechnern oder Clusterlösungen zu handhaben.

Deshalb sind mit der verfügbaren Personal-Computerhardware softwarebasierte dynamische Echtzeit Ray-Tracing Systeme nicht realisierbar. Die beschriebene Clusterlösung dürfte aus Kostengründen auf Spezialanwendungen beschränkt bleiben.

Demgegenüber liegt der vorliegenden Erfindung die Aufgabe zu Grunde, eine Vorrichtung vorzuschlagen, mit der sich Ray-Tracing-Verfahren schneller - vorzugsweise auch in Echtzeit - in dynamischen komplexen dreidimensionalen Szenen derart ausführen lassen, dass sich eine photorealistische Darstellung ergibt.

Diese Aufgabe wird durch eine Vorrichtung nach Anspruch 1 gelöst, indem diese Vorrichtung wenigstens einen programmierbaren Ray-Tracing Prozessor aufweist, in dem implementiert sind:

- spezielle Traversierungsbefehle und/oder
- Vektorarithmetikbefehle und/oder
- Befehle zur Erstellung von Ray-Tracing Beschleunigungsstrukturen und/oder
- wenigstens eine Entscheidungseinheit (Mailbox), mit der unterdrückt wird, dass bei Ausführung des Ray-Tracing-Verfahrens beim Schießen eines Strahles bereits mit dem Strahl geschnittene Objekte oder Dreiecke mehrfach mit dem Strahl geschnitten werden.

Der Aufbau der Vorrichtung ist so organisiert, dass mehrere Threads parallel abgearbeitet werden und mehrere Threads automatisch synchron abgearbeitet werden können.

Weiterhin verfügt die Vorrichtung über eine n-level Cache Hierarchie und / oder über ein virtuelles Speichermanagement und/oder eine direkte Verbindung mit dem Hauptspeicher.

Diese Vorrichtung kann vorzugsweise als FPGA und / oder in ASIC Technologie und / oder einer anderen logikbasierten Halbleitertechnologie oder in diskreter integrierter Logik, oder in der Kombination dieser Technologien realisiert sein.

Zur näheren Erörterung der Entscheidungseinheit wird auf Figur 2 verwiesen, aus der zu sehen ist, dass die Listen Einheit um eine Mailbox erweitert wird. Diese Mailbox verhindert, dass ein Dreieck oder Objekt beim Schießen eines Strahles mehrfach mit dem Strahl geschnitten wird, indem es sich bereits mit dem Strahl geschnittene Objekte oder Dreiecke merkt. Dadurch müssen nicht so viele Strahl-Objekt- bzw. Strahl-Dreiecks-Schnittpunktberechnungen durchgeführt werden, was die Berechnung beschleunigt. Die Mailbox kann als eine Art Schnittpunktberechnungscache angesehen werden, der im Gegensatz zu einem Speichercache nicht Speicheranfragen zum Speicher verhindert, sondern Schnittpunktberechnungen. Zur Implementierung der Mailbox können standardmäßige Cachingverfahren wie 4-Wege-Caches verwendet werden.

Anspruch 2 betrifft eine Vorrichtung zur photorealistischen Darstellung von dynamischen komplexen dreidimensionalen Szenen mittels des Ray-Tracing Verfahrens, wobei diese wenigstens eine spezielle Traversierungs-Einheit aufweist und wenigstens eine Listen-Einheit und wenigstens eine Entscheidungseinheit (Mailbox), mit der unterdrückt wird, dass bei Ausführung des Ray-Tracing-Verfahrens beim Schießen eines Strahles bereits mit dem Strahl geschnittene Objekte oder Dreiecke mehrfach mit dem Strahl geschnitten werden, und wenigstens eine Schnittpunktberechnungs-Einheit und wenigstens eine Einheit zum Erstellen von Beschleunigungsstrukturen und wenigstens eine Transformations-Einheit und/oder wenigstens eine Einheit zum Lösen von linearen Gleichungssystemen und dass mehrere Strahlen oder Threads parallel abgearbeitet werden können und mehrere Strahlen bzw. Threads automatisch synchron abgearbeitet werden können und beliebig viele Stufen von dynamischen Objekten in dynamischen Objekten realisiert werden können und dass die Vorrichtung über eine n-level Cache Hierarchie und / oder über ein virtuelles Speichermanagement und/oder eine direkte Verbindung mit dem Hauptspeicher verfügt.

Bei der Ausgestaltung nach Anspruch 3 ergibt sich als Unterschied zu der Ausgestaltung nach Anspruch 2, dass die wenigstens eine Transformations-Einheit und / oder die wenigstens eine Einheit zum Lösen von linearen Gleichungssystemen und / oder die wenigstens eine Schnittpunktberechnungs-Einheit im Anspruch 2 bei dem Anspruch 3

durch einen Ray-Tracing Prozessor ergänzt wurde, der bereits in Anspruch 1 erläutert wurde.

Diese auf dem Ray-Tracing Verfahren basierende Vorrichtung zur photorealistischen Darstellung dreidimensionaler bewegter Szenen, bei der die softwaremäßig definierten Beschleunigungsstrukturen und Verfahren in entsprechende Hardwarestrukturen umgesetzt sind, ist vorrangig zum Echtzeiteinsatz vorgesehen.

Zur Realisierung beliebiger ungeordneter Dynamik in einer Szene muss für jedes Bild der Bildfolge die Beschleunigungsstruktur neu berechnet werden. Dies bedeutet bei großen Szenen einen enorm großen Rechenaufwand da die gesamte Geometrie der Szene „angefasst" werden muss. Hierbei verschwindet der Vorteil der logarithmischen Komplexität in der Szenengröße.

Eine im Paper „A Simple and Practical Method for Interactive Ray-Tracing" beschriebene Lösung zu diesem Problem ist die Unterteilung der Szene in Objekte und ausschließlich das Bewegen dieser Objekte als Ganzes zu erlauben. Hierbei werden zwei Beschleunigungsstrukturen benötigt.

Eine Top-Level Beschleunigungsstruktur über den Objekten der Szene und jeweils eine Bottom-Level Beschleunigungsstruktur für jedes der Objekte. Die Objekte werden hierbei in Form von Instanzen von Objekten in der Szene positioniert.

Der Unterschied eines Objektes zu der Instanz desgleichen liegt darin, dass eine Instanz eines Objektes aus einem Objekt und einer Transformation besteht. Die Transformation ist eine affine Funktion, die das Objekt an eine beliebige Stelle der Szene verschiebt. Affine Transformationen erlauben des weiteren ein Skalieren, Rotieren und Scheren (engl. shearing) von Objekten. Im folgenden wird der Einfachheit halber auch für Instanzen von Objekten der Begriff Objekt verwendet, falls keine Verwechslungsmöglichkeit besteht.

Ein Strahl wird zunächst durch die Top-Level Beschleunigungsstruktur traversiert (Strahl durch die Szene verfolgen) bis ein mögliches Treffer-Objekt (das vom Strahl getroffene Objekt) gefunden wird. Nun transformiert man den Strahl in das lokale Koordinatensystem des Objektes und traversiert in der Bottom-Level Beschleunigungsstruktur des Objektes weiter bis ein Treffpunkt mit einem Primitiven Objekt gefunden ist. Primitive Objekte sind Objekte, die in sich keine weitere Struktur besitzen. Bei Ray-Tracern sind das in der Regel Dreiecke und Kugeln.

Diese Methode funktioniert in der Praxis sehr gut, jedoch nur so lange die Zahl der Objekte nicht zu groß wird, da die Top-Level Beschleunigungsstruktur in jedem Bild neu aufgebaut werden muss. Das neue Aufbauen dieser Beschleunigungsstruktur ist erforderlich, wenn die Objekte in dieser bewegt wurden.

Die Erfindung stellt nun eine Hardwarelösung dar, die obige Aufteilung der Szene in Objekte rekursiv unterstützt. Das heißt, sie schränkt sich nicht auf Objekte ein, die aus Primitiven Objekten bestehen, sondern erlaubt ebenfalls, dass sich diese Objekte wieder aus Objekten zusammensetzen, die wiederum aus Objekten bestehen können usw. Fig.1 zeigt wie aus mehreren Stufen von Objekten ein Baum erstellt werden kann. Zunächst wird als Objekt der Stufe 1 ein Blatt modelliert. Dieses Blatt wird nun mehrfach instantiiert und an einen Ast gesetzt, wodurch ein weiteres Objekt entsteht, jedoch jetzt ein Objekt der Stufe 2. Diese kleinen Äste können nun wieder mehrfach instantiiert werden zu einem größeren Ast oder Baum als Objekt der Stufe 3 usw. Es ist anzumerken, dass hier mehrere Ebenen von Objekten in Objekten vorkommen und dass die Repräsentation der Szene durch das mehrmalige Benutzen gleicher Geometrien klein ist.

Das in der Erfindung nach Anspruch 2 zum Einsatz kommende Verfahren für das Ray-Casting sieht wie folgt aus:

Der Strahl wird durch die Beschleunigungsstruktur der obersten Stufe traversiert bis ein mögliches Treffer-Objekt gefunden ist. Falls das Objekt ein Primitives Objekt ist, so wird der Schnittpunkt des Strahles mit dem Objekt berechnet. Ist das Objekt kein Primitives Objekt, so wird der Strahl in das lokale Koordinatensystem des Objektes transformiert und setzt dort die Traversierung rekursiv fort.

Ein wesentlicher Teil des Verfahrens ist die Transformation des Strahles in das lokale Koordinatensystem des Objektes, wodurch im Prinzip die Positionierung des Objektes durch die affine Transformation rückgängig gemacht wird. Das heißt, der transformierte Strahl sieht das Objekt nun nicht mehr transformiert. Dieser Transformationsschritt erfordert eine recht aufwändige affine Transformation des Strahlstartpunktes und der Strahlrichtung, wobei jedoch die dazu erforderliche aufwändige Hardwareeinheit zusätzlich noch für weitere Aufgaben einsetzbar ist. Es stellt sich heraus, dass die Transformationseinheit ebenfalls zur Berechnung des Schnittpunktes mit vielen Arten von Primitiven Objekten, zur Berechnung von Primärstrahlen und zur Berechnung vieler Arten von Sekundärstrahlen verwendet werden kann.

Zur Berechnung der Primärstrahlen wird eine ähnliche Kameratransformationsmatrix angewandt, wie bei den bekannten Rasterisierungsverfahren. Zunächst werden

Pre-Primärstrahlen der Gestalt $R=((0,0,0),(x,y,1))$, also Strahlen mit dem Startpunkt (0,0,0) und der Richtung (x,y,1) definiert, wobei x und y die Koordinaten des Pixels zu dem ein Primärstrahl berechnet werden soll darstellt. Zu jeder Kameraposition und Ausrichtung gibt es eine affine Transformation, die den Strahl R derart transformiert, dass er genau der Einfallsrichtung des Pixel (x,y) der Kamera entspricht.

Um den Schnittpunkt mit einem Primitiven Objekt zu berechnen, wird der Strahl in einen Raum transformiert, in dem das Primitive Objekt normiert ist. Im Falle eines Dreiecks als Primitives Objekt, wird der Strahl beispielsweise derart in einen Raum transformiert, dass das Dreieck die Gestalt $\Delta_{norm}=((1,0,0),(0,0,0),(0,1,0))$ hat. Zur Veranschaulichung wird auf Figur 10 verwiesen. Diese Transformation kann durch eine affine Transformation geschehen. Die anschließende Schnittpunktberechnung mit dem Normdreieck ist im Gegensatz zum allgemeinen Fall sehr einfach in Hardware zu lösen. Wird die Transformation derart gewählt, dass die Dreiecksnormale auf den Vektor (0,0,1) im Dreiecksraum transformiert werden, so lässt sich das Skalarprodukt aus Strahl und Dreiecksnormale sehr einfach im Dreiecksraum berechnen, da das Skalarprodukt aus Strahlrichtung (xt,yt,zt) und der Dreiecksnormalen (0,0,1) gerade 0*xt+0*yt+1*zt = zt ist.

Die Transformation kann des weiteren so gewählt werden, dass nur 9 Fließkommazahlen für deren Repräsentation benötigt werden, indem die Dreiecksnormale auf eine geeignete Normale im Norm-Dreiecksraum abgebildet wird. Dies verhindert jedoch die Möglichkeit, das Skalarprodukt im Norm-Dreiecksraum zu berechnen.

Es ist klar ersichtlich, dass diese Normobjekttransformation auch für andere Arten von Objekten verwendet werden kann, wie beispielsweise Kugeln, Ebenen, Quader, Zylinder und viele weitere geometrische Gebilde, es ist jeweils nur eine andere Schnittpunktberechnungseinheit zu erstellen.

Ein großer Vorteil hierbei ist die Tatsache, dass jede Art von Primitivem Objekt die gleiche Repräsentation im Speicher besitzt, nämlich eine affine Transformation, die in den Objektnormraum transformiert. Dies erleichtert die Konzeption des Speicherinterface einer Hardwarelösung. Die Transformation in den Objektnormraum nennt man die Normraumtransformation.

Schattenstrahlen sowie Spiegelungen lassen sich durch die Berechnung geeigneter Transformationen und geeigneter Strahlen effizient durch die Transformationseinheit berechnen.

Des weiteren ist es möglich, mit der Transformationseinheit Normalen (Vektoren die senkrecht auf einer Fläche stehen) zu transformieren. Diese Normalentransformation ist deswegen nötig, da einige Shadingmodelle die Normale der Geometrie am Auftreffpunkt benötigen. Diese Normale muss jedoch im Weltkoordinatensystem vorliegen, was bei obigem Verfahren nicht zwangsläufig der Fall ist. Vielmehr liegt die Normale erstmals nur im lokalen Koordinatensystem des getroffenen Objektes vor. Sie muss von dort wieder zurück in das Weltkoordinatensystem transformiert werden.

Die Transformationseinheit hat jedoch auch einen Nachteil. Da die affinen Transformationen, die als Matrizen gespeichert werden können, sowohl für Dreiecke also auch für die Objekte der Szene vorberechnet werden müssen, ist es nicht ohne weiteres möglich, die Position der Dreieckseckpunkte effizient von Frame zu Frame zu ändern. Dies ist in Vertexshadern auf heutigen Grafikkarten jedoch möglich. Vertexshader sind programmierbare Spezialeinheiten, die darauf optimiert sind, Bewegungen von Punkten im Raum zu berechnen.

Um dies zu ermöglichen, muss man sich von der Vorberechnung der Daten lösen. Demzufolge ist es dann erforderlich, zur Schnittpunktsberechnung mit einem Dreieck ein lineares Gleichungssystem mit drei Unbekannten zu lösen. Dieses explizite Lösen erfordert zwar mehr Fließkomma-Operationen, ist jedoch in Verbindung mit Vertexshadern erforderlich. Demzufolge kann es sinnvoll sein, obige Transformationseinheit durch eine Einheit, die ein lineares Gleichungssystem löst, zu ersetzen. Diese Einheit kann unter anderem dazu verwendet werden, um mit Drelecken zu schneiden oder Strahlen in das lokale Koordinatensystem eines Objektes zu transformieren.

Das in der Erfindung nach Anspruch 1 zum Einsatz kommende Verfahren für das Ray-Casting gliedert sich analog zu dem nach Anspruch 2 beschriebenen Verfahren, mit dem Unterschied, dass die Transformation und die Schnittpunktberechnung durch geeignete Anweisungen für den Ray-Tracing Prozessor umgesetzt werden. Hierbei bietet sich insbesondere die Möglichkeit alternative Verfahren zur Objekt-Strahl-Schnittpunktberechnung einzusetzen. Beispielsweise ermöglicht die Verwendung des Plücker-Strahl-Dreieckstests den effizienten Einsatz von Vertexshadern.

Ein Problem von detailreichen Szenen sind unerwünschte Aliasing Effekte, die vor allem dann entstehen, wenn die Objektdichte in einer Richtung sehr hoch ist. Dann kann es passieren, dass ein Strahl beispielsweise ein schwarzes Dreieck trifft und bei einer minimalen Bewegung der Kamera plötzlich ein weißes Dreieck getroffen wird. Solche

Effekte führen zu einem zeitlichen und örtlichen Rauschen im Bild. Der Grund liegt darin, dass Ray-Tracing in der Regel unendlich schmale Strahlen verwendet und nicht berücksichtigt, dass sich das Licht, das einen Pixel beeinflusst, pyramidenförmig ausbreitet und sich der Strahl mit der Entfernung aufweitet. So müssten eigentlich alle Objekte, die sich in dieser Strahlenpyramide befinden, zur Berechnung der Pixelfarbe herangezogen werden, was in einem Echtzeitsystem nicht möglich ist. Abhilfe schafft hier eine neue vereinfachte Form des Cone-Tracings. Anstatt einen beliebig schmalen Strahl zu betrachten, wird zusätzlich der Öffnungswinkel des Strahles bewertet. So kann je nach Entfernung zur Kamera die entsprechende Strahlbreite berechnet werden. Trifft man bei der Traversierung auf einen Teilraum, der zu einem Großteil von dem Strahl überdeckt wird, so ist es unter Umständen nicht sinnvoll, weiter zu traversieren. An dieser Stelle kann vorteilhaft eine vereinfachte Geometrie des Volumeninneren zur Berechnung verwendet werden. Dabei kann dann ignoriert werden, dass in dem Volumina vielleicht eine Million Dreiecke sind. Diese Dreiecke bilden unter Umständen nur die Wand eines Gebirges, die wegen der Größe des Strahles auch durch eine farbige Ebene approximiert werden kann.

Falls die Dreiecke jedoch ein löchriges Gebilde wie zum Beispiel den Eifelturm modellieren, ist als Approximation eher die Farbe der konstruktiven Gitterelemente und ein Transparenzwert zu wählen.

Vorteilhaft lassen sich solche vereinfachten Geometrierepräsentationen in der Beschleunigungsstruktur unterstützen. Figur 6 zeigt das Konzept am Beispiel eines Octrees. Zu dem fett umrandeten Volumina, zu dem ein Knoten in der Beschleunigungsstruktur gehört, ist die vereinfachte Geometrie abgebildet. Der Strahl überlappt fast mit dem gesamten Volumen des Knotens, so dass die vereinfachte Geometrie zur Schnittpunktsberechnung verwendet wird.

Eine alternative Methode besteht darin, die Objekte der Szene mit unterschiedlichen Detaillevel abzuspeichern. Das heißt, dass die Objekte mit unterschiedlicher Auflösung oder Anzahl von Dreiecke modelliert werden und dass weiterhin abhängig von der Entfernung der Objekte zur Kamera detailreiche oder vereinfachte Objekte verwendet werden.

Als nachteilig könnte bei der oben beschriebenen festverdrahteten, hardwarebasierten Ray-Tracing-Pipeline angesehen werden, dass es aufwändig ist, sie programmierbar zu gestalten. Im Vergleich zu einem Software Ray-Tracing Ansatz, wirkt die hardwarebasierte Pipeline sehr starr und speziell.

Abhilfe schafft die Entwicklung einer speziell auf das Ray-Tracing Verfahren angepassten CPU. Dieser spezielle Ray-Tracing Prozessor besteht aus einer Standard CPU, wie beispielsweise einem RISC Prozessor, dessen Befehlssatz um spezielle Befehle erweitert wird. Insbesondere ist ein Traversierungs-Befehl wichtig, der den Strahl durch eine Beschleunigungsstruktur traversiert. Einige Stellen des Verfahrens erfordern des weiteren aufwändige arithmetische Operationen, die vorwiegend im dreidimensionalen Raum geschehen. Es ist demnach sinnvoll, die CPU mit einer Vektorarithmetikeinheit auszustatten, ähnlich den heute geläufigen SSE2 Befehlssätzen.

Eine weitere Optimierung der CPU kann dadurch erreicht werden, dass die Parallelisierbarkeit des Verfahrens ausgenutzt wird. Es ist demnach möglich, sehr effektiv mehrere Threads (Programmabläufe) auf einer CPU laufen zu lassen, was die Auslastung und Effektivität der CPU deutlich erhöht. Dies gilt vor allem im Hinblick auf die Speicherwartezeiten. Macht ein Thread eine Speicheranfrage, so kann ein anderer während der Anfrage ausgeführt werden. Ein beispielhafter Aufbau solch einer CPU ist in Figur 5 zu sehen.

Da bei dynamischen Szenen für jedes Frame Neuberechnungen von Beschleunigungsstrukturen durchgeführt werden müssen, ist es erforderlich, den Befehlssatz der CPU um spezielle Befehle zur Erstellung von Beschleunigungsstrukturen zu erweitern. Bei der Erstellung von Beschleunigungsstrukturen muss oft entschieden werden, ob sich ein Objekt in einem gegebenen Teilraum befindet, in den es einsortiert werden soll, oder nicht. Eine spezielle Einheit, welche die Erstellung von Beschleunigungsstrukturen optimiert, kann die erforderliche Berechnung dadurch beschleunigen, dass diese eine sehr einfach zu berechnende Vorabentscheidung fällt. Als Teilräume werden häufig Boxen verwendet, deren 6 Begrenzungsflächen senkrecht auf der x,y bzw z- Achse stehen. Solch eine Box kann durch ihre Eckpunkte charakterisiert werden, wobei die Definition von lediglich zwei Punkten sogar ausreicht. Die Bestimmung/Entscheidung, ob sich das Dreieck in dieser Box befindet, kann in vielen Fällen mittels simpler Vergleiche der Koordinaten der Punkte erfolgen. Liegt das Dreieck zum Beispiel in X-Richtung weit links von der Box so sind die X – Koordinaten der 3 Eckpunkte des Dreiecks alle kleiner als die kleinste X- Koordinate der Eckpunkte der Box (siehe Fig. 13). Auch viele andere Konstellationen können so entschieden werden, zum Beispiel, ob sich das Dreieck vollständig in der Box befindet. Falls keine Entscheidung möglich ist, so müssen aufwändige mathematische Formeln wie das SAP (Separating Axis Theorem) angewendet werden. Die Entscheidung, ob eine Box mit einer anderen Box überlappt, kann ebenfalls mit den Eckpunktvergleichen entschieden werden.

Zur weiteren Optimierung wird eine Entscheidungseinheit eingesetzt, die verhindert, dass beim Schießen eines Strahles bereits mit dem Strahl geschnittene Objekte oder Dreiecke mehrfach mit dem Strahl geschnitten werden. Dies geschieht dadurch, dass die Listen Einheit wie in Figur 2 zu sehen um eine Mailbox erweitert wird. Diese Mailbox verhindert, dass ein Dreieck oder Objekt beim Schießen eines Strahles mehrfach mit dem Strahl geschnitten wird, indem es sich bereits mit dem Strahl geschnittene Objekte oder Dreiecke merkt. Dadurch müssen nicht so viele Strahl-Objekt- bzw. Strahl-Dreiecks-Schnittpunktberechnungen durchgeführt werden, was die Berechnung beschleunigt. Die Mailbox kann als eine Art Schnittpunktberechnungscache angesehen werden, der im Gegensatz zu einem Speichercache nicht Speicheranfragen zum Speicher verhindert, sondern Schnittpunktberechnungen. Zur Implementierung der Mailbox können standardmäßige Cachingverfahren wie 4-Wege-Caches verwendet werden.

Die Einheiten der Ray-Tracing Architektur benötigen eine sehr hohe Speicherbandbreite, das heißt, es müssen sehr viele Daten pro Zeiteinheit übertragen werden. Normalerweise ist dies nur zu realisieren, indem sehr viele Speicherchips parallel geschaltet werden. Die erforderliche Speicherbandbreite kann jedoch auch durch eine geeignete Verschaltung von mehreren Cachestufen (n-Level Caches) sichergestellt werden. Wesentlich ist hier eine Eigenschaft des Ray-Tracing Verfahrens, die als Kohärenz bezeichnet wird. Kohärenz bezeichnet die Tatsache, dass Strahlen, die ähnliche Bereiche des 3D Raumes durchlaufen, auch auf nahezu die gleichen Daten in der Beschleunigungsstruktur zugreifen und dementsprechend also auch auf die gleichen Objekte. Wird diese Eigenschaft ausgenutzt, können hohe Cache Hitraten erzielt werden. Das heißt, die benötigten Daten werden mit großer Wahrscheinlichkeit im Cache wiedergefunden und brauchen nicht zeitaufwändig aus dem Hauptspeicher geladen zu werden. Die Caches an sich sind entsprechend der Darstellung in Figur 4 beispielsweise in einem binären Baum angeordnet, um mehrere Ray-Tracing Einheiten zu versorgen.

Die erfindungsgemäße Vorrichtung kann in Verbindung mit einem 3D Display natürlich auch zur photorealistischen dreidimensionalen Echtzeitdarstellung komplexer bewegter Szenen verwandt werden. Abhängig von der Technologie des eingesetzten Displays sind hierbei drei Ausführungsformen der Bildausgabe zu unterscheiden.

Erstens eine Ausführungsform, bei der abwechselnd zwei den Stereoeindruck beinhaltende Bilder horizontal versetzt im Zeitmultiplexverfahren auf einem Display dargestellt werden.

Zweitens eine Ausführungsform, bei der zwei den Stereoeindruck repräsentierende, horizontal versetzte Bilder, die in abwechselnden senkrechten, die Bildinformation der beiden Bilder beinhaltende Streifen auf einem Display dargestellt werden.

Drittens eine Ausführungsform, bei der die beiden horizontal versetzten Bilder auf zwei getrennten Displays gleichzeitig oder im Zeitmultiplexverfahren dargestellt werden.

Die beiden horizontal versetzten Bilder, die jeweils dem rechten oder linken Auge zuzuordnen sind, werden durch entsprechend räumliche Displayanordnungen oder durch den Einsatz von Bildtrennvorrichtungen (z.B. Shutterbrillen, streifenförmige Fresnel-Prismen/Linsen, Polarisationsfilter) jeweils nur einem Auge sichtbar. Die einzusetzenden 3D Displays und deren Erfordernisse der Videosignalansteuerung entsprechen dem Stand der Technik und werden nicht näher beschrieben. Weitere Ausführungen zum Stand der Technik von 3D Displays sind folgenden, beispielhaft genannten Schriften zu entnehmen : Computer Graphics / Addison-Wesley ISBN 0201848406, DE 4331715, DE 4417664, DE 19753040, DE 19827590, DE 19737449

Der Einsatz computeranimierter photorealistischer Echtzeitdarstellung dreidimensionaler bewegter Szenen und Bilder erstreckt sich über die Darstellung dreidimensionaler CAD Daten, die Darstellung medizinischer und technisch- analytischer Daten, über die Filmanimation sowie dem Einsatz in Flug- und Fahrsimulatoren, bis zu den sogenannten Home Anwendungen in Computerspielen mit aufwändiger Echtzeitgraphik.

Darüber hinaus können die gleichen Verfahren ohne weitere Änderungen an der funktionalen Ausführung auch zur nicht-photorealistischen Bilderzeugung (z.B. Strichzeichnungen oder Comic-Stil Darstellungen) verwendet werden. Ebenso sind ohne weitere technische Änderungen Berechnungen durchführbar, die üblicherweise nicht direkt mit Bildberechnung in Verbindung gebracht werden. Beispielsweise gehören hierzu die Kollisionserkennung zwischen geometrischen Objekten (engl. Collision Detection) und das diskrete Lösen von numerischen Problemen. Alle beschriebenen Anwendungen sind nicht auf den interaktiven Bereich beschränkt und können ohne Änderungen an Verfahren oder Vorrichtung auch im Offline-Bereich, beispielsweise bei der Berechnung von Kinofilmen oder sehr aufwändigen physikalischen Simulationen, eingesetzt werden.

Die funktionelle Realisierung und die Hardwareimplementierung der Ray-Tracing Verfahren erfolgt in komplexen und schnellen Logiktechnologien, wobei deren Umsetzung sowohl als festverdrahtete Digital-Logik in Form von diskreter Digital-Logik bestehend aus einzelnen Standard IC, oder Kunden- oder Anwendungsspezifisch gefertigten IC beispielsweise ASIC, oder von komplexen programmierbaren Logikbausteinen /

Logikschaltkreisen, beispielsweise CPLD oder FPGA Technologien mit oder ohne CPU Kern, oder der Kombination dieser Technologien erfolgt.

Die nachfolgend beschriebene, beispielhafte Ausführungsform der Erfindung beschreibt die Ray-Tracing Einheit einer Computer-Graphikkarte, bei der die Hardwareimplementierung der Ray-Tracing Verfahren beispielsweise in einem freiprogrammierbaren Logikbaustein FPGA, in ASIC Technologie, oder in einem festverdrahteten Spezial Chip erfolgen kann.

Soweit Verfahren und Funktionsabläufe beschrieben werden, so sind diese rein hardwaremäßig zu realisieren. Das bedeutet, dass entsprechende Logikeinheiten und hardwaremäßig realisierte Arithmetikeinheiten zu gestalten sind.

Die Standardfunktion der Ansteuerelektronik zur Ansteuerung des Datendisplays (Kathodenstrahlröhre, TFT-, LCD- oder Plasmamonitor) und deren Timing entsprechen dem Stand der Technik, werden als bekannt vorausgesetzt und sind nicht Gegenstand der Beschreibung. Eine Schnittstelle zwischen dem Bildspeicher dieser Standardfunktion und der erfindungsgemäßen Umsetzung der Ray-Tracing Verfahren wird beschrieben.

Die Beschreibung gliedert sich in zwei Teile. Zunächst wird die Ray-Casting-Pipeline abgekürzt RCP beschrieben. Dabei handelt es sich um den Kern des Designs, der Strahlen durch die Szene traversiert und den Auftreffpunkt zurückliefert.

Im zweiten Teil wird für die Ray-Casting-Pipeline eine optimierte Ray-Tracing Architektur beschrieben, in der mehrere dieser Ray-Casting-Pipelines zusammenarbeiten.

Die Figur 2 zeigt die Ray-Casting-Pipeline Einheit (RCP), die aus mehreren Untereinheiten besteht. Die Traversierungs-Einheit traversiert den Strahl durch eine geeignete Beschleunigungsstruktur, vorzugsweise einen k-D Baum. Der Strahl wird so lange traversiert, bis er in einen Bereich der Szene kommt, in dem sich mögliche Treffer-Objekte befinden. Die Objekte dieses Bereiches sind in einer Liste gespeichert, die von der Listen-Einheit bearbeitet wird. Die Listen-Einheit beinhaltet auch eine Mailbox, die verhindert, dass ein Dreieck oder Objekt beim Schießen eines Strahles mehrfach mit dem Strahl geschnitten wird, indem es sich bereits mit dem Strahl geschnittene Objekte oder Dreiecke merkt. Dadurch müssen nicht so viele Strahl-Objekt- bzw. Strahl-Dreiecks-Schnittpunktberechnungen durchgeführt werden, was die Berechnung beschleunigt.

Diese Objekte müssen nun auf einen möglichen Schnittpunkt hin untersucht werden. Falls es keinen gültigen Schnittpunkt gibt, muss mit der Traversierung fortgesetzt werden. Die

Listen-Einheit sendet die möglichen Treffer-Objekte, die nicht bereits bearbeitet wurden, eins nach dem anderen, zur Matrix-Lade-Einheit, die die zu dem Objekt gehörige affine Transformation lädt. Diese affine Transformation kann durch eine 4x3 Matrix dargestellt werden. Es kann sich dabei um Objektraumtransformationsmatrizen oder um Matrizen handeln, die in den Normraum eines Primitiven Objektes transformieren. Nachdem die Matrix-Lade-Einheit die Matrix in der Transformationseinheit gespeichert hat, werden die Strahlen von der Strahl-Einheit durch die Transformationseinheit geschickt.

Nach der Transformation sind nun zwei Szenarien möglich. Zum einen kann es sich um ein Objekt handeln, das noch weitere Objekte beinhaltet. Ist dies der Fall, so wandern die Strahlen wieder zurück in die Traversierungs-Einheit und der transformierte Strahl wird in dem Objekt weiter traversiert. Handelt es sich jedoch um ein Primitives Objekt, so geht der Strahl direkt weiter in die Schnittpunktberechnungs-Einheit, die den Stahl mit dem Normobjekt schneidet. Die Schnittpunktberechnungs-Einheit kann wie zuvor beschrieben mehrere Normobjekte (Dreiecke, Kugeln usw.) unterstützen.

Die berechneten Schnittpunkt-Daten werden in der Schnittpunkt-Einheit gesammelt. Die Schnittpunkt-Einheit liefert einen Rückkanal zur Traversierungs-Einheit, so dass diese erkennen kann, ob schon gültige Schnittpunkt-Daten vorhanden sind.

Die Traversierungs-Einheit, Listen-Einheit und Matrix-Lade-Einheit sind die einzigen Einheiten der Ray-Casting-Pipeline, die auf externen Speicher zugreifen. Die Traversierungs-Einheit greift auf die Beschleunigungsstruktur zu, die Listen-Einheit auf Listen von Objektadressen und die Matrix-Lade-Einheit auf affine Transformationen in Form von Matrizen. Alle drei Einheiten sind über einen eigenen Cache mit dem Hauptspeicher verbunden, um die nötige Speicherbandbreite zu gewährleisten.

Eine vereinfachte Version der Ray-Casting-Pipeline ist in Figur 12 dargestellt, wobei nur die wichtigsten Einheiten abgebildet sind: die Traversierungs-Einheit, welche die Strahlen durch die Beschleunigungsstruktur traversiert, die Listen-Einheit, welche die Listen bearbeitet, die Transformations-Einheit, welche die geladene Transformation auf die Strahlen anwendet, und die Schnittpunktberechnungs-Einheit, die den transformierten Strahl mit dem Norm-Objekt schneidet.

Die Ray-Casting-Pipeline ist entsprechend der Darstellung der Figur 3 in eine geeignete Ray-Tracing Architektur eingebettet. Die Abbildung zeigt 4 Ray-Casting-Pipeline Einheiten mit ihren jeweils 3 Caches. Es sind hier jeweils 2 Einheiten mit einer Shading Einheit verbunden. Diese Shading Einheiten verwenden die Ray-Casting-Pipeline Einheiten, um die Farben der Pixel des Bildes zu berechnen. Hierzu schießt die Shading Einheit

Primärstrahlen, verarbeitet die Auftreffinformationen, die die Ray-Casting-Pipeline zurückliefert, und schießt Sekundärstrahlen beispielsweise zu Lichtquellen. Bei den Shadingeinheiten kann es sich um festverdrahtete Hardware oder um die später beschriebenen programmierbaren Ray-Tracing Prozessoren handeln.

Die Shading Einheiten besitzen einen Kanal zur Transformationseinheit der Ray-Casting-Pipeline. Dieser wird verwendet, um Kameramatrizen und Matrizen für Sekundärstrahlen zu laden und dadurch den Rechenaufwand für die Shading Einheit zu minimieren. Es ist vorteilhaft, wenn die Shading Einheiten jeweils einen getrennten Textur Cache und Shading Cache besitzen. Der Shading Cache enthält Shadinginformationen zu der Geometrie der Szene wie zum Beispiel Farben und Materialdaten. Der Texturcache ist mit dem Texturspeicher verbunden und ermöglicht den Shading Einheiten Zugriff auf Texturen. Es ist vorteilhaft, wenn jede Shading Einheit ihren eigenen lokalen Frame-Buffer besitzt, auf dem sie die Farben und Helligkeitswerte der gerade bearbeiteten / berechneten Pixel ablegt. Des weitern ist es von Vorteil, falls ein Z-Buffer vorhanden ist, da dieser für die nachfolgend beschriebene Anbindung an die Standard-Rasterisierungshardware benötigt wird.

Ist die Farbe eines Pixels von der Shading Einheit vollständig berechnet, so kann diese Farbe über die optionale Tone-Mapping-Einheit in den globalen Frame-Buffer geschrieben werden. Die Tone-Mapping-Einheit wendet eine einfache Funktion auf die Farbe an, um diese in dem 24 Bit RGB Raum abzubilden. Auch die geometrischen Tiefenwerte (Z-Werte), die im optionalen lokalen Z-Buffer gespeichert sind, können nun in den optionalen globalen Z-Buffer übertragen werden.

Die Farbe bzw. der neue Z-Wert werden jedoch nur dann in den Frame-Buffer bzw. Z-Buffer geschrieben, wenn der zuvor im Z-Buffer vorhandene Z-Wert größer ist. Hierdurch ist sichergestellt, dass nur dann Pixel geschrieben werden, wenn sie geometrisch vor bereits von der Rasterisierungshardware oder anderen Ray-Tracing Passes berechneten Werten liegen. So ist es möglich, die Ray-Tracing Hardware mit einer Standard Rasterisierungshardware zu kombinieren, die auf dem gleichen Frame-Buffer bzw. Z-Buffer arbeitet, der hierbei auch die Schnittstelle zu dieser Standard Rasterisierungshardware darstellt.

Zur weiteren Erhöhung der Systemleistung können optional noch weitere Shading-Einheiten mit den zugehörigen Ray-Casting-Pipelines und Caches parallel geschaltet werden. Der leistungssteigernde Effekt liegt hierbei in der Verbreiterung der Daten- und Verarbeitungsstruktur.

Es ist nicht zu erwarten, dass die gesamte Szene in den lokalen Hauptspeicher eines Ray- Tracing Chips passt. Um dieses Problem zu lösen, kann der lokale Hauptspeicher als ein großer Cache verwendet werden, der größere Blöcke der Szene cached. Die eigentliche Szene befindet sich an anderer Stelle und wird über DMA bei Bedarf nachgeladen. Dieses Verfahren zur virtuellen Speicherbehandlung ermöglicht es, sehr große Szenen zu visualisieren, die nicht in den Hauptspeicher des Ray-Tracing Chips passen.

Photon Mapping ist ein Standardverfahren, indem virtuelle Photonen von den Lichtquellen aus in die Szene geschossen und auf den Flächen der Szene angesammelt werden. Die Lichtverteilung der Szene kann somit simuliert werden. Dies gilt vor allem auch für Kaustiken. Werden Photonen geschossen, so wird eine Beschleunigungsstruktur über den Photonen aufgebaut, die beispielsweise ein k-d Baum sein kann. Nun kann ein Abbild dieser berechneten Photonenlichtverteilung erfolgen, indem die Szene mit standardmäßigen Ray-Tracing Verfahren visualisiert wird und an jedem Auftreffpunkt die dort eintreffende Lichtintensität derart in die Farbberechnung einfließt, dass die Energie aller in der Nähe dieses Punktes auftreffenden Photonen aufsummiert wird. Hierzu müssen alle benachbarten Photonen in der Beschleunigungsstruktur der Photonen gesucht werden. Diese Aufgabe kann von der Traversierungseinheit unterstützt werden, indem nicht an einem Strahl entlang, sondern ein Volumina traversiert wird. Auf diese Weise können alle benachbarten Photonen bearbeitet werden, indem beispielsweise ihre Energie aufsummiert wird.

In der Regel arbeiten Ray-Tracing Verfahren mit unendlich schmalen Strahlen, was zu Samplingartefakten (Aliasing) führt. Eine viel bessere Approximation ergibt sich durch die Verwendung von Strahlkegeln anstelle von Strahlen. Das Licht, das ein Pixel der Kamera beleuchtet, kommt nicht nur aus einer diskreten Richtung, sondern aus einer Art Pyramide, die durch einen Kegel sehr gut approximiert werden kann. Das Traversieren dieses Raumvoluminas von vorne nach hinten, beschrieben durch einen Strahlkegel oder eine Strahlpyramide, kann ebenfalls von der Traversierungseinheit als Spezialfunktion durchgeführt werden. Figur 6 zeigt eine zweidimensionale Abbildung eines Strahlkegels.

Die Signalaufbereitung zur Ansteuerung des Displays und die Timinggenerierung für das Display bzw. den Monitor kann in bekannter Art durch die optionale Rasterisierungshardware erfolgen oder durch geeignete funktionale Einheiten umgesetzt werden falls vorteilhaft für die gewünschte Anwendung. So können die Grundfunktionen der Standard Rasterisierungshardware mit den hardwareimplementierten Ray-Tracing

Verfahren und Funktionen zu einer sehr leistungsfähigen Echtzeit Hardwarearchitektur verbunden werden.

Eine zweite, weitere beispielhafte Ausführungsform der Erfindung basiert auf der Gestaltung und Anwendung von frei programmierbaren Ray-Tracing CPUs oder Ray-Tracing Prozessoren, die programmgesteuert die erfindungsgemäß beschriebenen, speziellen Ray-Tracing Funktionen und Verfahren ausführen. Hierbei werden durch entsprechende Logik- und Funktionsparallelität jeweils nur wenige, vorzugsweise ein oder zwei CPU Taktzyklen, zur Abarbeitung der Einzelfunktion benötigt.

Soweit interne Algorithmen, Verfahren und Funktionsabläufe der Ray-Tracing Prozessoren, beschrieben werden, sind diese mittels einer Hardware-beschreibungssprache, zum Beispiel HDL, VHDL, oder JHDL zu erstellen und auf die entsprechende Hardware zu übertragen. Die Hardwareimplementierung der Ray-Tracing Prozessoren mit den implementierten Verfahren und Funktionen kann beispielsweise in einem freiprogrammierbaren Logikbaustein FPGA, in ASIC Technologie, in der Kombination von digitalen Signalprozessoren mit FPGA / ASIC oder in einem festverdrahteten Spezial Chip erfolgen.

Dies bedeutet, dass entsprechende Logikeinheiten und hardwaremäßig realisierte Arithmetikeinheiten zu gestalten sind, deren Einzelfunktionen programmgesteuert abgerufen werden können. Abhängig von der Komplexität der eingesetzten Hardwaretechnologie können pro Chip eine oder mehrere Ray-Tracing Prozessoren, mit oder ohne weiteren Logikfunktionen realisiert werden.

Ray-Tracing Prozessoren sind voll programmierbare Recheneinheiten, die auf die Ausführung von Vektorarithmetikbefehlen und speziellen Ray-Tracing Befehlen wie "Traversieren" und "Erstellen von Beschleunigungsstrukturen" ausgelegt sind. Hierbei kann eine Auslegung unter Hinzunahme von Funktionseinheiten erfolgen oder aber auch mit bereits vorhandenen Funktionseinheiten unter Hinzunahme von ggf. wenigen Logikbausteinen. Beispielsweise kann das Traversieren durch spezielle Funktionseinheiten erfolgen oder durch eine Erweiterung der vorhandenen arithmetischen Funktionseinheiten um wenige logische Funktionseinheiten.

Wie in Figur 11 zu sehen ist werden mehrere der in Figur 5 dargestellten Ray-Tracing Prozessoren parallel geschaltet. Das Speicherinterface wird von einer Cache-Hierarchie gebildet, welche die nötige Speicherbandbreite bereitstellt. Dieses Vorgehen ist wegen der starken Kohärenz benachbarter Strahlen effizient möglich. Bei dem Ray-Tracing Prozessor Konzept bearbeitet jeder Ray-Tracing Prozessor entweder genau ein Pixel des

Bildes oder ein Paket von mehreren Pixeln. Hierbei entspricht die Berechnung jedes einzelnen Pixels einem Berechnungsthreads. Mehrere dieser Threads werden zu einem Paket gebündelt und als ganzes synchronisiert abgearbeitet. Synchron abgearbeitete Threads zeichnen sich dadurch aus, dass diese immer gemeinsam die gleiche Instruktion ausführen. Dies ermöglicht die Erstellung effizienter Hardware und die Durchführung von Speicheranfragen pro ganzem Paket und nicht einzeln für jeden Thread. Vor allem diese Reduktion der Speicheranfragen ist ein wesentlicher Vorteil der Bearbeitung von Paketen von Threads.

Der Thread Generator von Figur 11 erstellt Threads, die auf den Ray-Tracing Prozessoren ausgeführt werden. Der Thread Generator kann auch von den Ray-Tracing Prozessoren programmiert werden. Spezielle Funktionen zum Abtasten der Pixel des Bilder auf eine Cache-kohärente Art und Weise (zum Beispiel Hilbert Kurve) erlaubt es, die Ray-Tracing Prozessoren optimal mit kohärenten Threads zu versorgen. Dies reduziert die benötigte Speicherbandbreite. Der Thread Generator hat auch Zugriff über ein DMA-Interface auf den Hauptspeicher. Es können also auch die Startwerte für die einzelnen Threads in einem vorherigen Bearbeitungsschritt erstellt und in den Speicher geschrieben werden, um später aus diesen Daten wieder neue Threads zu generieren.

Das Bearbeiten eines Pixels geschieht durch ein Softwareprogramm, das auf dem Ray-Tracing Prozessor läuft. Dieses Softwareprogramm beschreibt einen rekursiven Verfahrensablauf zum Berechnen des Farbwertes eines Pixels, selbst wenn die Hardware auf Paketen von Strahlen arbeitet. Das Paketmanagement ist also transparent zum Programmiermodell.

Figur 5 zeigt einen beispielhaften Aufbau eines Ray-Tracing Prozessors. Zu sehen ist ein Standard-Prozessorkern (RISC Kern), dem zwei Spezialcoprozessoren parallel geschaltet sind. Die Coprozessoren besitzen jeweils ihre eigenen Register. Es ist jedoch auch möglich, durch Spezialbefehle diese Registerinhalte von einem Coprozessor in einen anderen zu überführen. Der Traversierungs-Kern ist ein spezieller Coprozessor, der Strahlen effizient durch eine Beschleunigungsstruktur traversieren kann. Hierzu benötigt er ein spezielles Speicherinterface zu den Knoten der Beschleunigungsstruktur (Knoten-Cache). Der Vektor Arithmetik Kern ist ein Spezialcoprozessor, der effizient Operationen im 3D Raum durchführen kann. Die von jeder Ray-Tracing Software benötigten Vektoradditionen, Skalarmultiplikationen, Kreuzprodukte und Vektorprodukte können mittels dieser Einheit schnell berechnet werden. Die Vektorarithmetikeinheit benötigt Zugriff auf einen Spezialcache, der es ermöglicht, ganze Vektoren in einem Takt zu laden.

Die Semantik des Traversierungsbefehls könnte folgendermaßen aussehen. Die Ray-Tracing CPU beschreibt spezielle Register mit dem Ursprung und der Richtung des Strahles und der Adresse der Beschleunigungsstruktur, die traversiert werden soll. Nun wird ein spezieller Befehl aufgerufen, der eine spezielle Traversierungseinheit startet. Diese Einheit traversiert die Beschleunigungsstruktur und sendet alle Knoten, die Objekte enthalten, zu einer Listeneinheit, die einen Mailboxingmechanismus enthalten kann, der auch schon in der ersten beispielhaften Ausführungsform der Erfindung beschrieben wurde. Für jedes Objekt wird nun ein kleines Programm auf der CPU ausgeführt, das den Strahl mit diesem Objekt schneidet. Ist ein gültiger Schnittpunkt des Strahles mit einem Objekt gefunden, so wird zu dem Programm zurückgekehrt welches den Strahl geschossen hatte.

Die in dem Ray-Tracing Prozessor hardwaremäßig implementierten Einzelbefehle oder Logikfunktionen beinhalten die gleichen Algorithmen, die schon bei der festverdrahteten ersten Ausbildungsform der Erfindung beschrieben sind. Ergänzend hierzu kann dieser Befehlssatz jedoch um weitere festverdrahtete Befehle und Funktionen ergänzt werden, die wiederum programmgesteuert zu aktivieren sind. Durch spezielle Traversierungs-Operationen, durch Vektorarithmetik, sowie durch die parallele Abarbeitung mehrer Threads auf einem Ray-Tracing Prozessor wird die für Echtzeitanwendung erforderliche Rechenleistung bereitgestellt und gleichzeitig werden die Auswirkungen der Speicherlatenzen (Wartezeiten auf Speicheranfragen) auf die Systemgeschwindigkeit minimiert oder sogar irrelevant.

~~*Dadurch, dass die RTPU programmgesteuert dazu verwandt werden kann, rekursiv Sekundärstrahlen durch die Szene zu schießen, wird das Programmiermodell im Vergleich zur festverdrahteten Hardware erheblich vereinfacht.*~~

Ist die Farbe des Pixels vollständig berechnet, kann diese in den Frame-Buffer geschrieben werden. Gegebenenfalls kann zusätzlich die Distanz in den Z-Buffer geschrieben werden. Hierdurch ist eine Anbindung an Rasterisierungshardware und die entsprechende Darstellung auf einem Display möglich.

Anspruch 10 beschreibt die Verwendung einer zusätzlichen raumaufteilenden Datenstruktur, die keine geometrischen Objekte speichert oder referenziert, sondern Raumeinflüsse oder materialverändernde Parameter enthält. Solche Raumeinflüsse können beispielsweise Nebel, Dunst, Staubpartikel oder heißer Rauch sein, wobei beispielsweise heißer Rauch auch eine Veränderung der durch das Volumen des Rauchs sichtbaren Szene zur Folge haben kann. Weitere Raumeinflüsse sind Lichtquellen oder

materialverändernde Auren wie sie beispielsweise in der Darstellung von Fantasiewesen verwendet werden können. Hierbei erlaubt die Verwendung der raumaufteilenden Datenstruktur den Berechnungsaufwand erheblich zu reduzieren, da nur Einflüsse berücksichtigt werden müssen, die räumlich so gelegen sind, dass sie einen Einfluss haben können.

Anspruch 11 beschreibt eine Erweiterung, die es ermöglicht, dreidimensionale Szenen zu verarbeiten, die nicht bzw. nicht ausschließlich aus Dreiecken aufgebaut sind und die bei Bedarf andere geometrische Primitive in Dreiecke oder eine einfach zu verarbeitende Zwischendarstellung überführt. Hierbei wird die zusätzliche Funktionalität dadurch erreicht, dass die Hardware entweder um Funktionseinheiten erweitert wird oder vorhandene Einheiten die Funktionalität umsetzen.

Anspruch 12 beschreibt eine Erweiterung, die es ermöglicht, pro Strahl nicht nur den nächstgelegenen; sondern zusätzlich weitere Objekt-Strahl-Schnittpunkte als Ergebnis einer Strahlverarbeitung zu berechnen. Hierbei ist es vorteilhaft, wenn die Ergebnisse nach der Entfernung vom Strahlursprung sortiert sind. Die maximale Anzahl der Schnittpunkte pro Ergebnis kann als Konstante festgelegt werden oder pro Strahl in Abhängigkeit von Parametern der geschnittenen Objekte erfolgen. Diese Technik kann verwendet werden, um Strahlen durch transparente Objekte effizienter zu berechnen.

Anspruch 13 beschreibt eine Erweiterung, die mit zusätzlichen und/oder den vorhandenen Funktionseinheiten zählen kann, wie oft ein bestimmtes Element in den Berechnungen eines Bildes verwendet wurde. Hierbei können die zu zählenden Elemente sehr unterschiedlich sein und eine mögliche Klassifizierung der Elemente könnte über die Adresse im Speicher oder die ID eines Elementes erfolgen. Zu diesen Elementen gehören:

dynamische und geometrische Objekte, Teile oder vollständige Materialbeschreibungen, Elemente oder Teilgruppen einer Raumbeschreibungsdatenstruktur, Programme oder Programmfunktionen, einzelne Speicherzellen und ganze Speicherseiten oder -bereiche.

Anspruch 14 beschreibt eine Erweiterung der Funktionalität zum Berechnen der Raumbeschreibungsdatenstrukturen zu Teilen oder vollständigen dreidimensionalen Szenen, wobei zusätzliche Parameter pro dynamischem Objekt oder dynamischem Teilobjekt oder geometrischem Objekt einen Einfluss auf die Berechnungsweise der Raumbeschreibungsdatenstruktur haben. Solche Einflüsse können beispielsweise sein, dass ein Objekt spezifiziert, dass es in bestimmten Raumvolumina keine geometrischen

Objekte besitzt. Dies erlaubt es, die Raumbeschreibungsdatenstruktur effizienter zu berechnen und zusätzlich den Berechnungsaufwand pro Bild weiter zu reduzieren.

Beschreibung der Zeichnungen

Figur 1 zeigt, wie aus mehreren Stufen von Objekten ein Baum erstellt werden kann. Zunächst wird als Objekt 101 der Stufe 1 ein Blatt modelliert. Dieses Blatt 101 wird nun mehrfach instantiiert und an einen Ast 102 gesetzt, wodurch ein weiteres Objekt entsteht, jedoch jetzt ein Objekt der Stufe 2. Diese kleinen Äste 102 können nun wieder mehrfach instantiiert werden zu einem Baum 103 als Objekt der Stufe 3.

Figur 2 zeigt die Ray-Casting-Pipeline 200 mit den wichtigsten Datenpfaden, sowie der Schnittstelle zu den Caches. Der Pfeil 201 geht zur Shading-Einheit, der Pfeil 202 kommt von der Shading-Einheit. Weiterhin bedeuten:

203: Traversierungs-Einheit
204: Listen-Einheit welche eine Mailbox enthält
205: Matrix-Lade-Einheit
206: Strahl-Einheit
207: Transformations-Einheit
208: Schnittpunktberechnungs-Einheit
209: Schnittpunkt-Einheit
210: Knoten Cache
211: Listen Cache
212: Matrix Cache

Figur 3 zeigt das Top-Level Diagramm einer beispielhaften Implementierung der Erfindung. Die 4 Ray-Casting-Pipelines (RCP) sind über eine Cache-Hierarchie mit den getrennten Speichern für Knoten, Listen und Matrizen verbunden. In diesem Beispieldesign sind je zwei RCP's mit einer Shadingeinheit verbunden, die Zugriff auf einen lokalen Frame-Buffer und Z-Buffer hat. Über diese werden Farbwerte mittels einer Tone-Mapping-Einheit in einen globalen Frame-Buffer geschrieben und Tiefenwerte direkt in einen globalen Z-Buffer geschickt. An diesen Z-Buffer und Frame-Buffer kann eine dem Stand der Technik entsprechende Rasterisierungshardware (Rasterisierungs Pipeline) angeschlossen werden.

Dabei bedeuten:

301: Rasterisierungshardware
302: Frame Buffer
303: Video Out
304: Tone Mapping Einheit
305: Z-Buffer
306: Frame Buffer 1
307: Z-Buffer 1

308: Textur-Cache 1
309: Shading Cache 1
310: Shading Einheit 1
311: Knoten Cache 1
312: Listen Cache 1
313: Matrix Cache 1
314: RCP 1
315 Knoten Cache 2
316: Listen Cache 2
317: Matrix Cache 2
318: RCP 2
319: Frame Buffer 2
320: Z-Buffer 2
321: Textur-Cache 2
322: Shading Cache 2
323: Shading Einheit 2
324: Knoten Cache 3
325: Listen Cache 3
326: Matrix Cache 3
327: RCP 3
328 Knoten Cache 4
329: Listen Cache 4
330: Matrix Cache 4
331: RCP 4

Figur 4 zeigt die Cache-Infrastruktur, welche die nötige interne Speicherbandbreite im Chip bereitstellt. In der Abbildung handelt es sich um einen binären n-stufigen Cache, es sind jedoch auch andere hierarchische Strukturen denkbar.

Dabei bedeuten:

401: Knoten Cache 1
402: Knoten Cache 2
403: Knoten Cache 3
404: Knoten Cache 4
405: Knoten Cache 5
406: Knoten Cache 6
407: Knoten Cache 7
408: Knoten Speicher
409: Listen Cache 1
410: Listen Cache 2
411: Listen Cache 3
412: Listen Cache 4
413: Listen Cache 5
414: Listen Cache 6
415: Listen Cache 7
416: Listen Speicher
417: Matrix Cache 1
418: Matrix Cache 2
419: Matrix Cache 3
420: Matrix Cache 4
421: Matrix Cache 5
422: Matrix Cache 6

423: Matrix Cache 7
424: Matrix Speicher
425: Texture Cache 1
426: Texture Cache 2
427: Texture Speicher
428: Shading Cache 1
429: Shading Cache 2
430: Shading Cache 3
431: Shading Speicher

Figur 5 zeigt die beispielhafte Ausführungsform einer Ray-Tracing CPU. Dabei haben die Blöcke folgende Bedeutung:

501: Instruktion Laden
502: Befehlsspeicher
503: RISC Kern
504: Cache
505: Traversierungs Kern
506: Knoten Cache
507: Vektor Arithmetik Kern
508: Vektor Cache

Figur 6 zeigt ein Beispiel der vereinfachten Geometrie in den Octreeknoten. Dabei ist mit der Bezugsziffer 601 der Strahlkegel bezeichnet und mit der Bezugsziffer 602 eine "Vereinfachte Geomtrie".

Figur 7 zeigt ein Beispiel der regelmäßigen Grid Beschleunigungsstruktur an einer einfachen Szene. Der Raum ist der Einfachheit halber nur in 2D gezeichnet. Mit der Bezugsziffer 701 ist der Strahl bezeichnet.

Figur 8 zeigt ein Beispiel der k-D Baum Beschleunigungsstruktur an einer einfachen Szene. Der Raum ist der Einfachheit halber nur in 2D gezeichnet. Mit der Bezugsziffer 801 ist der Strahl bezeichnet.

Figur 9 zeigt ein Beispiel der Octree Beschleunigungsstruktur an einer einfachen Szene. Der Raum ist der Einfachheit halber nur in 2D gezeichnet. Mit der Bezugsziffer 901 ist der Strahl bezeichnet.

Figur 10 zeigt die Dreieckstransformation aus dem Globalen Raum in den Norm-Dreiecks Raum. Es ist ein Strahl 1001 zu sehen sowie ein Dreieck 1002. Durch eine affine Transformation, die durch den Pfeil 1003 repräsentiert wird, entsteht ein transformierter Strahl 1004 sowie ein Normdreieck 1005. Der Globale Raum wird durch das linke Koordinatensystem repräsentiert und der Norm-Dreiecksraum durch das rechte Koordinatensystem.

Figur 11 zeigt eine beispielhafte Ausführungsform der Erfindung basierend auf speziellen Ray-Tracing Prozessoren. Dabei haben die Blöcke folgende Bedeutung:

1101: Rasterisierungshardware
1102: Thread Generator
1103: Z-Buffer / Frame Buffer
1104: Video - Out
1105: Ray-Tracing Prozessor 1
1106: Ray-Tracing Prozessor 2
1107: Ray-Tracing Prozessor 3
1108: Ray-Tracing Prozessor 4
1109: Cache 1
1110: Cache 2
1111: Cache 3
1112: Cache 4
1113: Cache 5
1114: Cache 6
1115: Cache 7
1116: Hauptspeicher

Figur 12 zeigt eine vereinfachte Version der in Figur 2 gezeigten Ray-Casting-Pipeline. Dabei haben die Blöcke folgende Bedeutung:

1201: Traversierungs-Einheit
1202: Knoten Cache
1203: Listen-Einheit
1204: Listen Cache
1205: Transformations-Einheit
1206: Matrix-Cache
1207: Schnittpunktberechnungs-Einheit

Der Pfeil 1208 kommt von der Shading-Einheit

Figur 13 zeigt einen einfachen Fall, in dem durch das Auswerten Vergleichen der Eckpunkte eines Dreiecks und der Eckpunkte einer Box entschieden werden kann, ob das Dreieck mit der Box überlappt oder nicht. Falls die X-Koordinaten des Dreiecks kleiner sind als die kleinste X-Koordinate der Box, so liegt das Dreieck außerhalb der Box. Das Dreieck ist mit der Bezugsziffer 1301 bezeichnet, die Box mit der Bezugsziffer 1302.

Patentansprüche

1. Vorrichtung zur photorealistischen Darstellung von dynamischen komplexen dreidimensionalen Szenen mittels des Ray-Tracing Verfahrens, dadurch gekennzeichnet, dass diese wenigstens einen programmierbaren Ray-Tracing Prozessor aufweist, in dem implementiert sind:

- spezielle Traversierungsbefehle und/oder
- Vektorarithmetikbefehle und/oder
- Befehle zur Erstellung von Ray-Tracing Beschleunigungsstrukturen und/oder
- wenigstens eine Entscheidungseinheit (Mailbox), mit der unterdrückt wird, dass bei Ausführung des Ray-Tracing-Verfahrens beim Schießen eines Strahles bereits mit dem Strahl geschnittene Objekte oder Dreiecke mehrfach mit dem Strahl geschnitten werden,

und dass die Vorrichtung so organisiert ist, dass mehrere Threads parallel abgearbeitet werden können und mehrere Threads automatisch synchron abgearbeitet werden können und dass die Vorrichtung über eine n-level Cache Hierarchie und/oder über ein virtuelles Speichermanagement verfügt und/oder über eine direkte Verbindung mit dem Hauptspeicher.

2. Vorrichtung zur photorealistischen Darstellung von dynamischen komplexen dreidimensionalen Szenen mittels des Ray-Tracing Verfahrens, dadurch gekennzeichnet, dass diese wenigstens eine spezielle Traversierungs-Einheit aufweist und wenigstens eine Listen-Einheit und wenigstens eine Entscheidungseinheit (Mailbox), mit der unterdrückt wird, dass bei Ausführung des Ray-Tracing-Verfahrens beim Schießen eines Strahles bereits mit dem Strahl geschnittene Objekte oder Dreiecke mehrfach mit dem Strahl geschnitten werden, und wenigstens eine Schnittpunktberechnungs-Einheit und wenigstens eine Einheit zum Erstellen von Beschleunigungsstrukturen und wenigstens eine Transformations-Einheit und/oder wenigstens eine Einheit zum Lösen von linearen Gleichungssystemen und dass mehrere Strahlen oder Threads parallel abgearbeitet werden können und mehrere Strahlen bzw. Threads automatisch synchron abgearbeitet werden können und beliebig viele Stufen von dynamischen Objekten in dynamischen Objekten realisiert werden können und dass die Vorrichtung über eine n-level Cache Hierarchie und/oder über ein virtuelles Speichermanagement und/oder über eine direkte Verbindung mit dem Hauptspeicher verfügt.

3. Vorrichtung nach Anspruch 1, dadurch gekennzeichnet, dass diese wenigstens eine spezielle Traversierungs-Einheit aufweist und wenigstens eine Listen-Einheit und

wenigstens eine Entscheidungseinheit (Mailbox), mit der unterdrückt wird, dass bei Ausführung des Ray-Tracing-Verfahrens beim Schießen eines Strahles bereits mit dem Strahl geschnittene Objekte oder Dreiecke mehrfach mit dem Strahl geschnitten werden, und wenigstens eine Schnittpunktberechnungs-Einheit und wenigstens eine Einheit zum Erstellen von Beschleunigungsstrukturen und wenigstens einen Ray-Tracing Prozessor.

4. Vorrichtung gemäß den Ansprüchen 1, 2 oder 3, dadurch gekennzeichnet, dass die wenigstens eine Einheit zum Erstellen von Beschleunigungsstrukturen durch spezielle Hardware oder durch programmierbare Einheiten oder Ray-Tracing Prozessoren realisiert wird und funktionell Verfahren durchführt zum Datenstrukturaufbau der Beschleunigungsstruktur sowie der Entscheidung, ob ein Dreieck oder eine Box eine andere Box überlappt, indem die wenigstens eine Einheit eine Entscheidung abhängig von Eckpunktvergleichen der Ecken des Dreiecks bzw. der Box und der Ecken der zweiten Box durchführt und - falls keine Entscheidung möglich ist - eine konservative Entscheidung trifft oder in diesem Fall ein Programm auf dem programmierbaren Ray-Tracing Prozessor gestartet wird, das die exakte Entscheidung fällt, oder eine weitere spezielle Hardwareeinheit die exakte Entscheidung fällt oder die ganze Berechnung auf dem Ray-Tracing Prozessor stattfindet.

5. Vorrichtung gemäß den Ansprüchen 3 oder 4, dadurch gekennzeichnet, dass die wenigstens eine Transformations-Einheit und/oder die wenigstens eine Logikeinheit zum Lösen linearer Gleichungssysteme funktionell zur Primärstrahlgenerierung und / oder Objektraumtransformation und / oder Dreiecks-Normraumtransformation und / oder Reflektionsstrahlberechnung und / oder Transparenzstrahlberechnung und / oder Schattenstrahlberechnung und / oder Normalentransformation verwendet wird.

6. Vorrichtung gemäß einem der Ansprüche 1 bis 5, dadurch gekennzeichnet, dass die wenigstens eine Traversierungseinheit bzw. der Traversierungsbefehl nicht nur entlang eines Strahles traversieren kann, sondern auch ein Volumina, so dass alle Objekte in diesem Volumina bearbeitet werden können.

7. Vorrichtung gemäß einem der Ansprüche 1 bis 6, dadurch gekennzeichnet, dass die wenigstens eine Traversierungseinheit bzw. der Traversierungsbefehl nicht nur entlang eines Strahles traversieren kann, sondern auch entlang eines Strahlkegels oder einer Strahlpyramide, so dass alle Objekte, die sich in dem Strahlkegel oder der Strahlpyramide befinden, von vorne nach hinten bearbeitet werden können.

8. Vorrichtung gemäß einem der Ansprüche 1 bis 7, dadurch gekennzeichnet, dass die Funktion der wenigstens einen Traversierungs-Einheit und die hardwaremäßige

Umsetzung der Traversierungs-Befehle darauf beruht, dass ein Strahl durch eine Beschleunigungsstruktur traversiert wird, die auf dem k-D Baum-Verfahren oder dem Octreeverfahren oder dem regelmäßigen Grid-Verfahren oder dem Bounding-Volume-Hierarchie Verfahren basiert, wobei in jedem Beschleunigungsstruktur-Knoten vereinfachte Geometriedaten gespeichert werden, die verwendet werden sobald der betrachtete Strahlkegel einen Großteil des zu diesem Knoten gehörigen Voluminas durchläuft.

9. Vorrichtung nach einem der Ansprüche 1 bis 6, dadurch gekennzeichnet, dass mehrere Ray-Tracing Einheiten auf mehreren Chips und/oder mehreren Platinen parallel arbeiten.

10. Vorrichtung nach einem der Ansprüche 1 bis 9, dadurch gekennzeichnet, dass die beschriebene Ray-Tracing Hardware zusätzlich eine raumaufteilende Datenstruktur verwendet, in der Raumeinflüsse und/oder materialverändernde Parameter gespeichert werden, die mit den vorhandenen und/oder zusätzlichen Funktionseinheiten ausgewertet werden.

11. Vorrichtung nach einem der Ansprüche 1 bis 10, dadurch gekennzeichnet, dass die Ray-Tracing Hardware dreidimensionale Szenen verarbeitet, die nicht ausschließlich aus Dreiecken aufgebaut sind sondern auch andere geometrische Objekte enthalten, die bei Bedarf in andere geometrische Objekte überführt und/oder direkt mit zusätzlichen und/oder vorhandenen Funktionseinheiten und/oder dem programmierbaren Ray-Tracing Prozessor bearbeitet werden.

12. Vorrichtung nach einem der Ansprüche 1 bis 11, dadurch gekennzeichnet, dass die beschriebene Ray-Tracing Hardware dreidimensionale Szenen verarbeitet und für einen Strahl keinen, einen oder mehrere Objekt-Strahl-Schnittpunkte sortiert oder unsortiert nach der Entfernung berechnet, wobei die Anzahl der berechneten Objekt-Strahl-Schnittpunkte als Konstante festgelegt sein und/oder durch zusätzliche Parameter der Objekte beschrieben werden kann.

13. Vorrichtung nach einem der Ansprüche 1 bis 12, dadurch gekennzeichnet, dass die beschriebene Ray-Tracing Hardware mit zusätzlichen und/oder den vorhandenen Funktionseinheiten zählen kann, wie oft ein dynamisches und/oder geometrisches Objekt und/oder eine Materialbeschreibung und/oder ein Element und/oder eine Teilgruppe der Raumbeschreibungsdatenstruktur und/oder ein Programm und/oder eine Speicherzelle und/oder eine Speicherseite in den Berechnungen eines Bildes verwendet wurden.

14. Vorrichtung nach einem der Ansprüche 1 bis 13, dadurch gekennzeichnet, dass die beschriebene Ray-Tracing Hardware mit zusätzlichen und/oder den vorhandenen Funktionseinheiten Raumbeschreibungsdatenstrukturen zu Teilen oder vollständigen dreidimensionalen Szenen berechnen kann, wobei zusätzliche Parameter pro dynamischem Objekt und/oder dynamischem Teilobjekt und/oder geometrischem Objekt einen Einfluss auf die Berechnungsweise der Raumbeschreibungsdatenstruktur haben.

15. Vorrichtung nach einem der Ansprüche 1 bis 14, dadurch gekennzeichnet, dass die beschriebene Ray-Tracing Hardware über einen gemeinsamen Z-Buffer und Frame-Buffer an eine Rasterisierungshardware angeschlossen ist, die sich auf dem gleichen oder einem separaten Chip wie die Ray-Tracing Hardware befindet.

16. Vorrichtung nach einem der Ansprüche 1 bis 15, dadurch gekennzeichnet, dass mehrere Ray-Tracing Einheiten parallel arbeiten und die zur Berechnung benötigten Daten auf die Speicher dieser Ray-Tracing Einheiten verteilt sind, und bei Bedarf jeweils von der Einheit welche die benötigten Daten speichert nachgeladen wird.

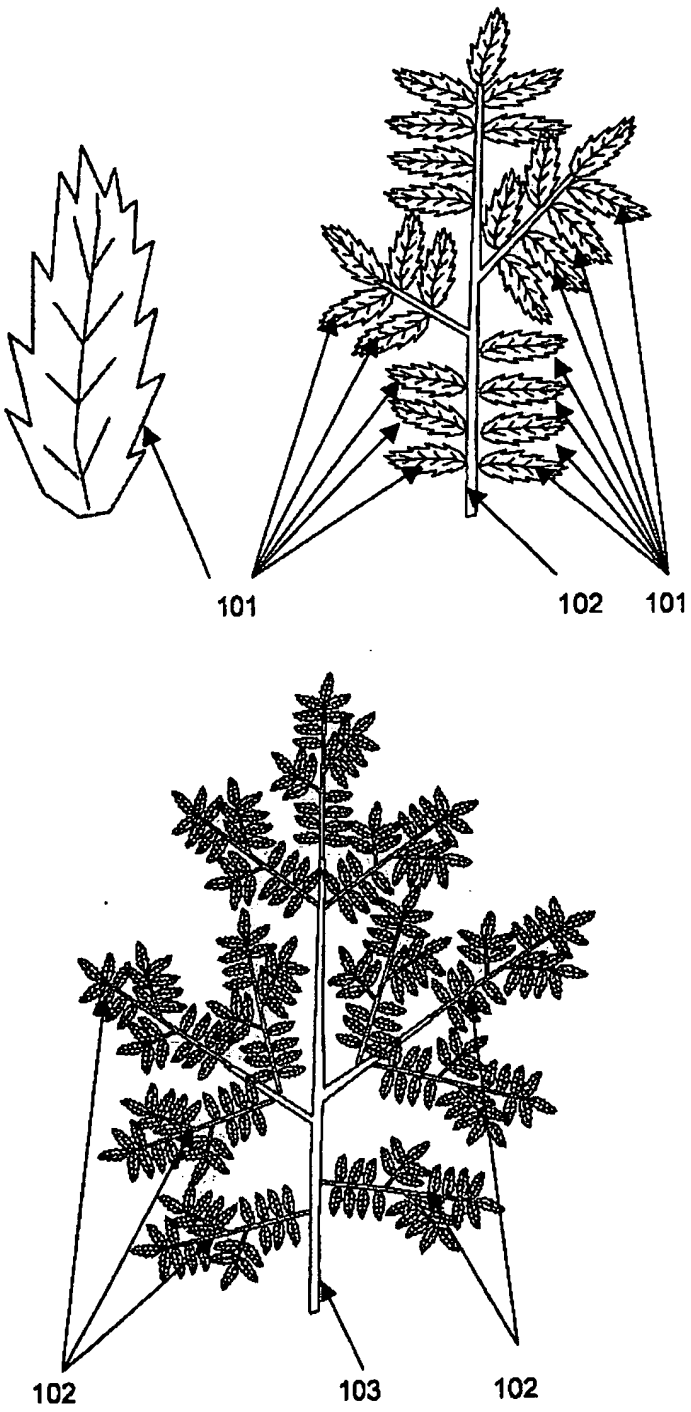

Fig. 1

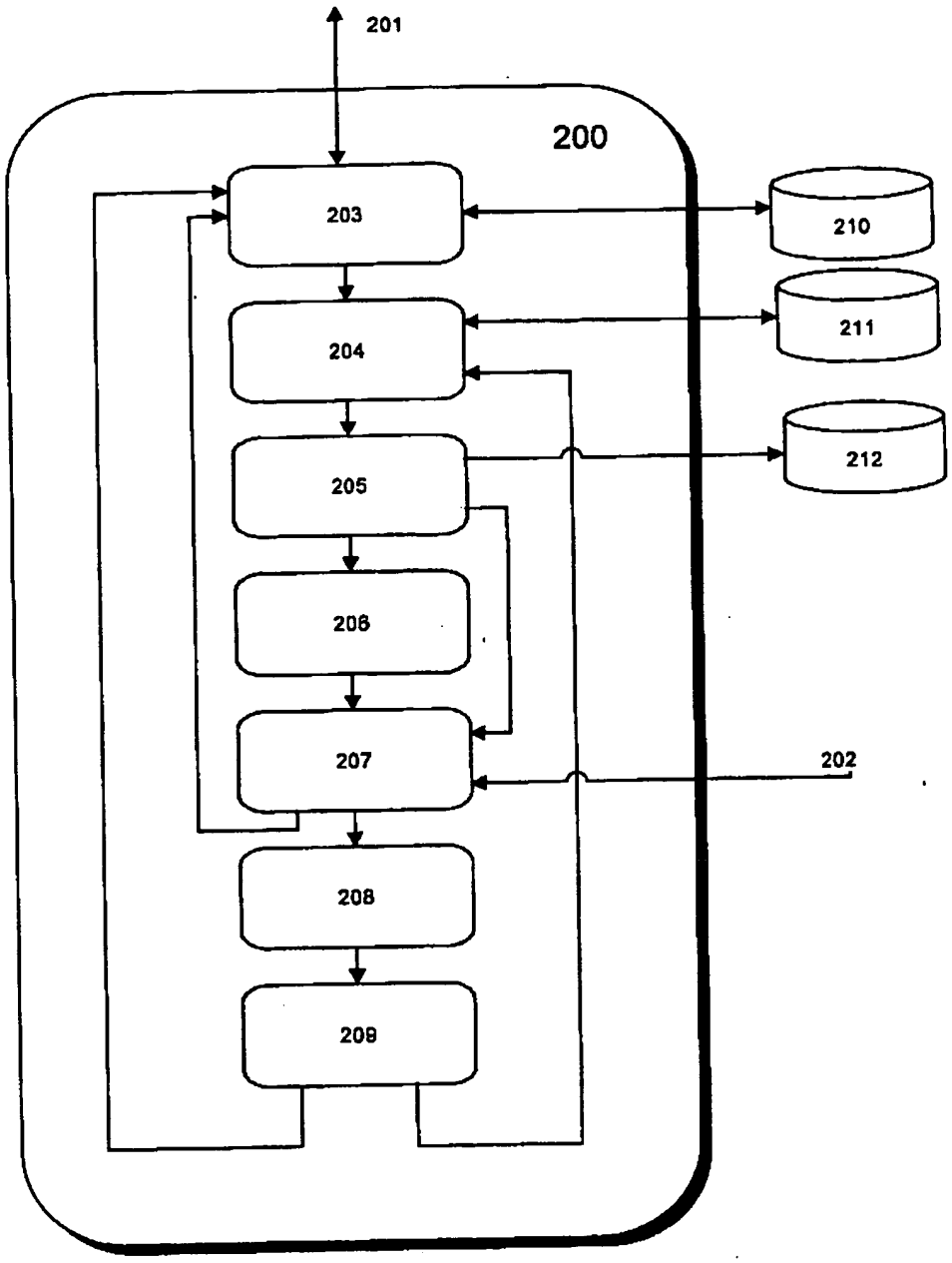

Fig. 2

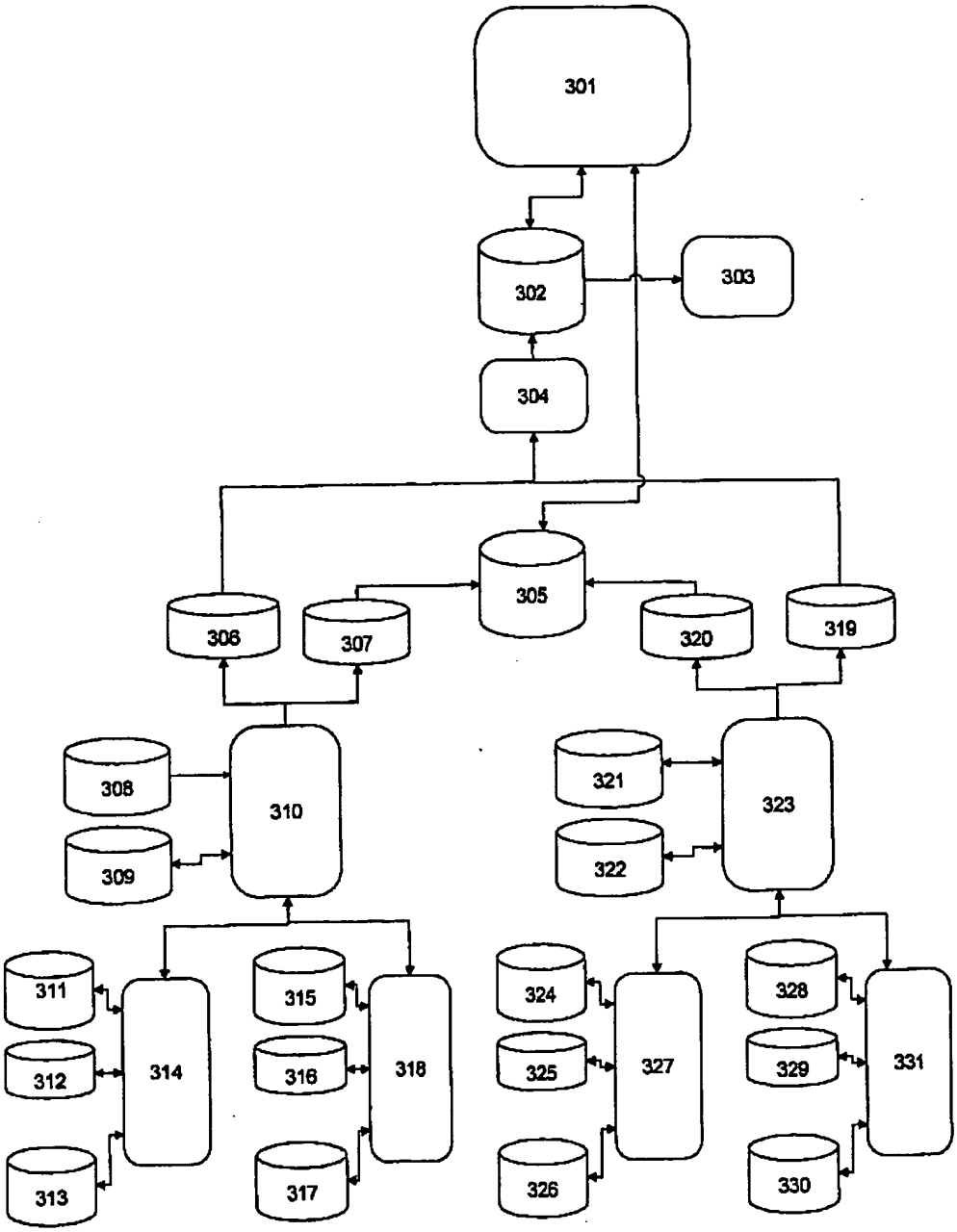
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331

Fig. 3

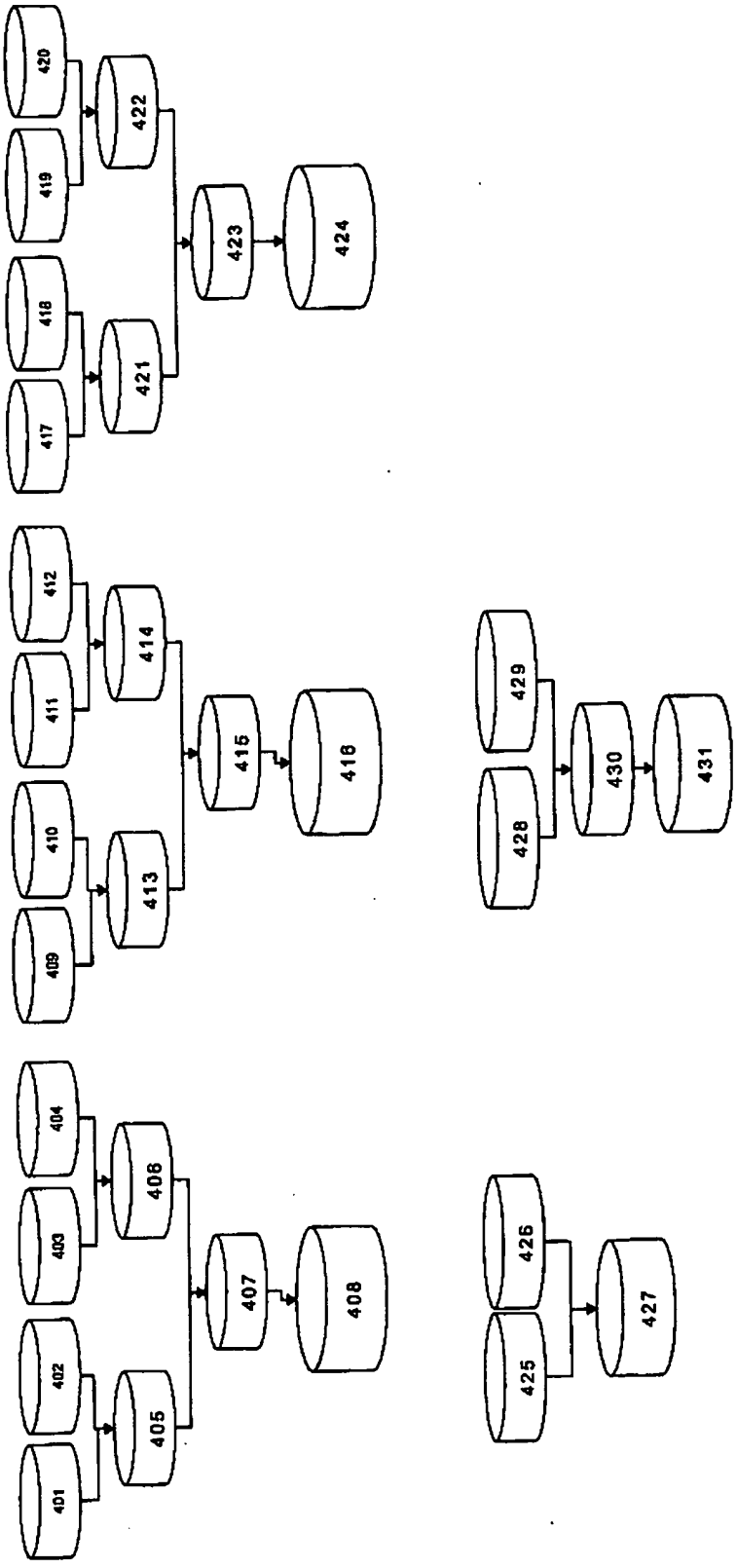
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431

Fig. 4

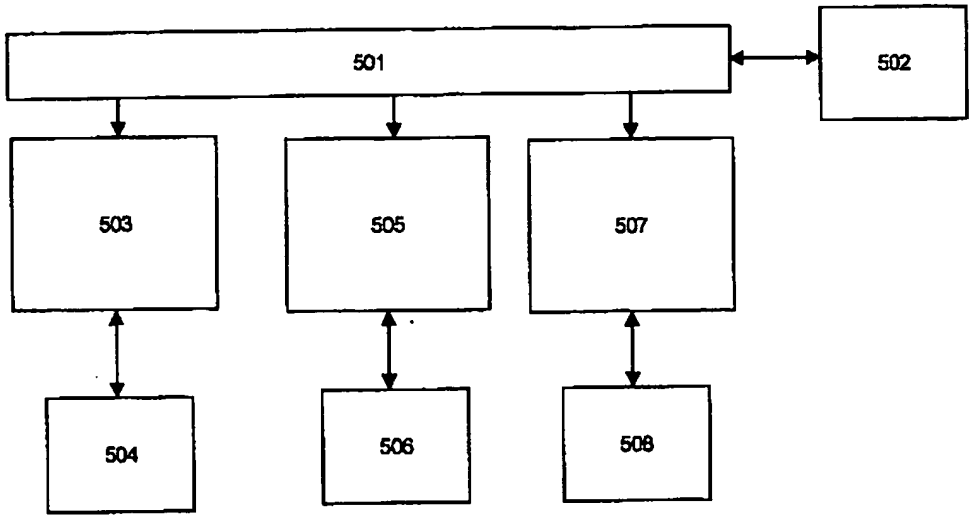

Fig. 5

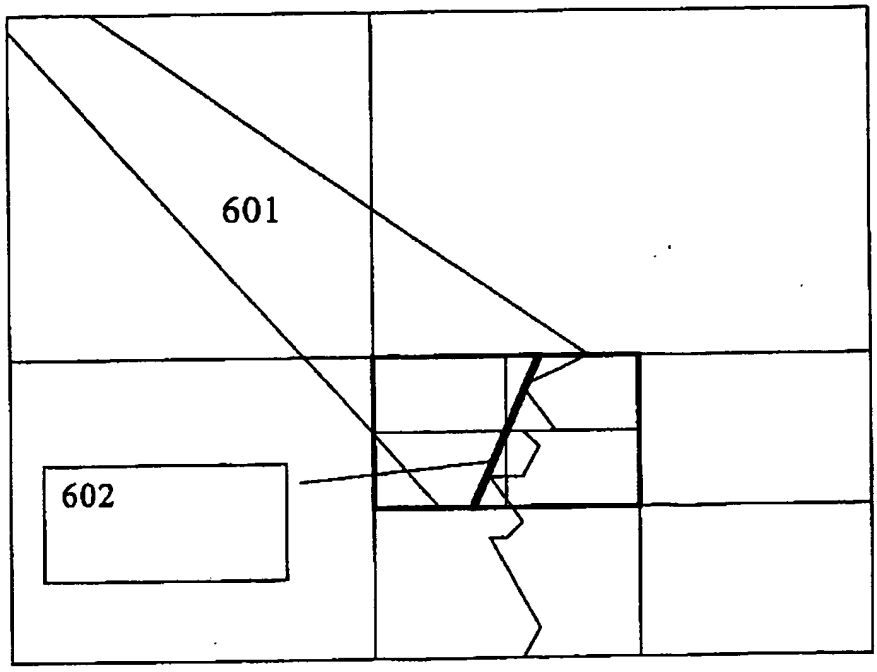

Fig. 6

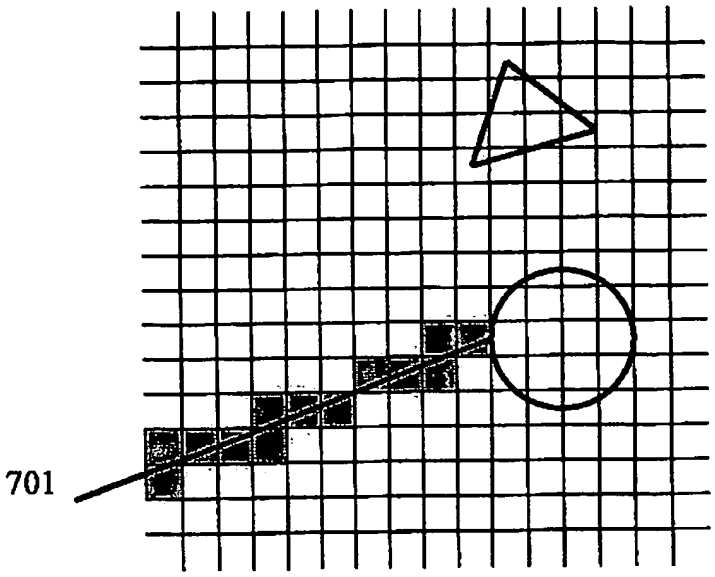

Fig. 7

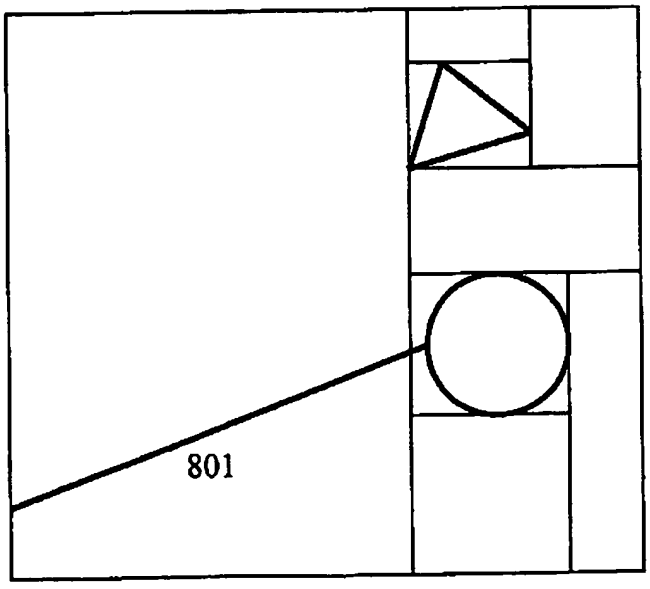

Fig. 8

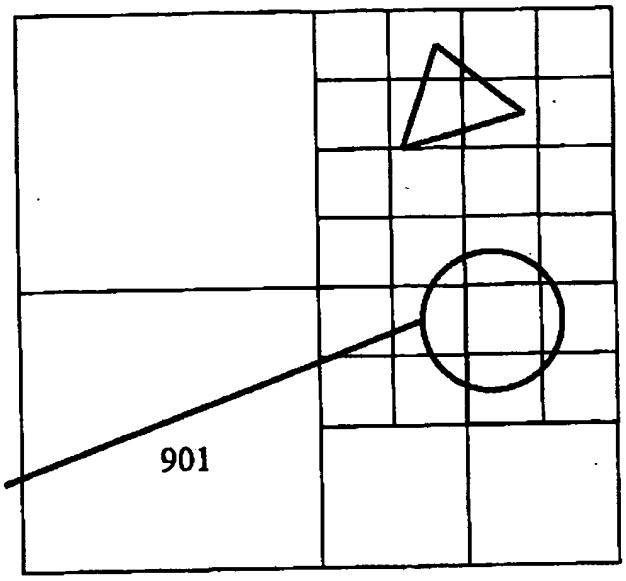

Fig. 9

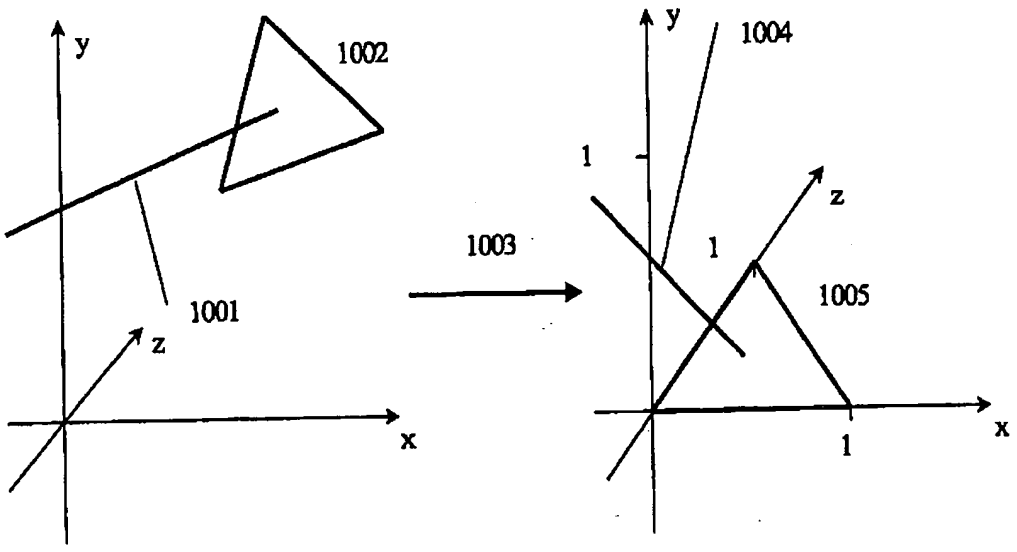

Fig. 10

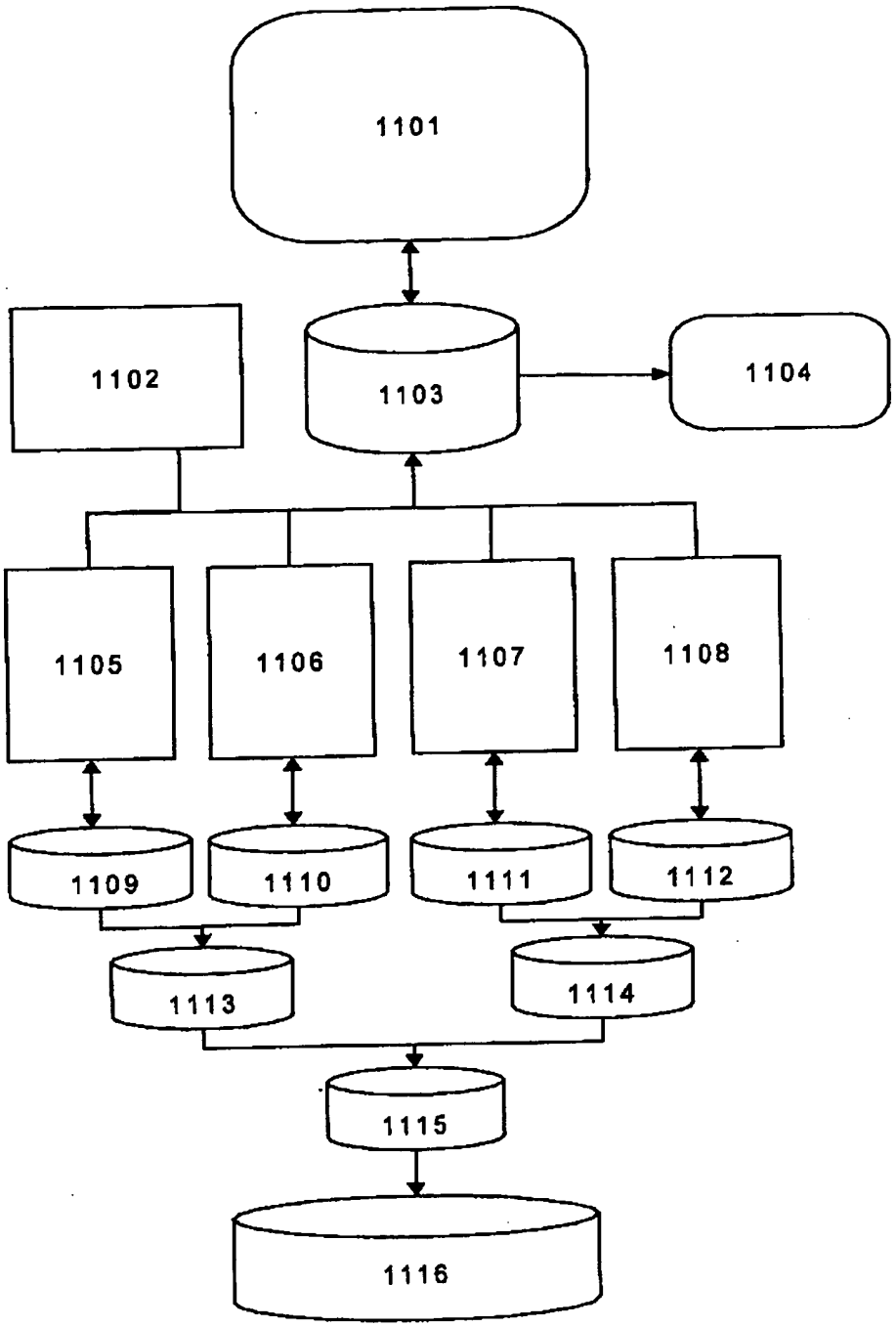
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116

Fig. 11

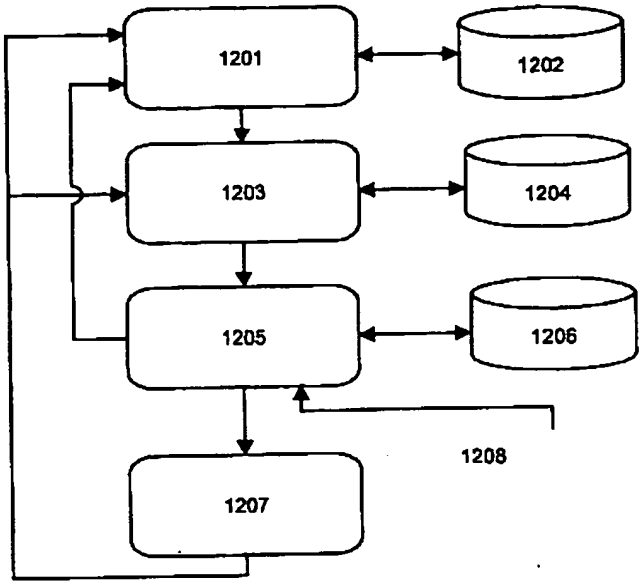

Fig. 12

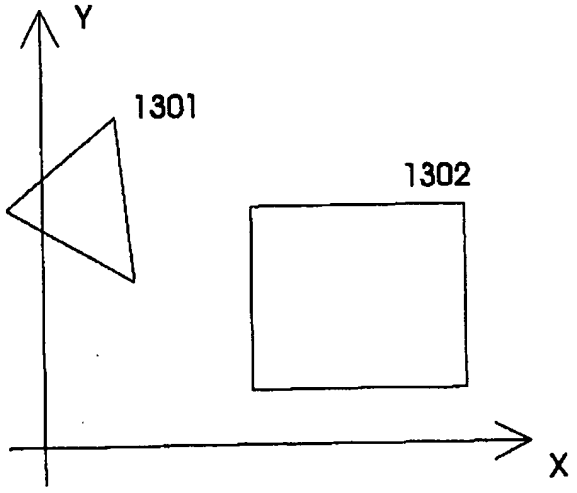

Fig. 13